НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ «КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра обчислювальної техніки

До захисту допущено:

Завідувач кафедри

Сергій СТІРЕНКО

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та мережі»

спеціальності 123 «Комп'ютерна інженерія»

на тему: «Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації»

Виконала:
студентка IV курсу, групи ІО-63
Світлана МОЖАРОВСЬКА _____

Керівник:
Доц. каф. ОТ, к. т. н.
Олександр МАРКОВСЬКИЙ _____

Консультант з нормоконтролю:
Професор, доктор технічних наук
Валерій СІМОНЕНКО _____

Рецензент:

_____

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студентка _____

Київ – 2020 року

Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського» Факультет інформатики та обчислювальної техніки Кафедра обчислювальної техніки

Рівень вищої освіти – перший (бакалаврський) Спеціальність – 123 «Комп'ютерна інженерія» Освітньо-професійна програма «Комп'ютерні системи та мережі»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Сергій СТІРЕНКО
«___»_____2020 р.

ЗАВДАННЯ
на дипломний проєкт студентці
Світлані МОЖАРОВСЬКІЙ

1. Тема проекту «Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації», керівник проєкту Марковський Олександр Петрович, доцент, к. т. н., затверджені наказом по університету від «07» травня 2020 р. №1081-с

2. Термін подання студентом проєкту 04.06.2020

3. Вихідні дані до проєкту технічна документація

4. Зміст пояснювальної записки
Аналіз проблеми ефективності ідентифікації віддалених абонентів в сучасних умовах. Огляд нині існуючих методів ідентифікації. Розробка способу використання технологій штучного інтелекту для криптографічно строгої ідентифікації користувачів. Розробка програмних засобів ідентифікації з використанням персептрону

5. Перелік графічного матеріалу

Схема ресурсів системи, схема роботи програми, діаграма класів, схема алгоритму.

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Норм. контроль	Валерій СІМОНЕНКО		

7. Дата видачі завдання 01.09.2019

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	*Затвердження теми роботи*	01.09.2019	
2.	*Вивчення та аналіз завдання*	02.09.2019-13.04.2020	
3.	*Розробка архітектури та загальної структури системи*	13.04.2020-04.05.2020	
4.	*Програмна реалізація системи*	04.05.2020-15.05.2020	
5.	*Оформлення пояснювальної записки*	13.04.2020-25.05.2020	
6.	*Захист програмного продукту*	25.05.2020	
7.	*Передзахист*	26.05.2020	
8.	*Захист*		

Студент Світлана МОЖАРОВСЬКА

Керівник Олександр МАРКОВСЬКИЙ

Анотація

Бакалаврський дипломний проєкт присвячений розробці програмних засобів моделювання застосування нейронних мереж в системах захисту інформації.

На основі аналізу задачі ідентифікації віддалених користувачів в рамках концепції нульових знань обґрунтовано вибір типу персептрона та організації програмних засобів моделювання його роботи. Розроблені програмні засоби моделювання використання багатошарового персептрона для криптографічно строгої ідентифікації віддалених користувачів.

Розроблена програма для одного із протоколів розглянутої схеми мовою Java, створено абстрактну модель функціонування системи ідентифікації.

Annotation

Bachelor's project is devoted to the development of modeling software tools applying to a neural network in information security systems.

Due to the analysis of remote users' identification issue considering zero informing conception there was substantiated process of choosing the right type of the perceptron and organized modeling software tools of its work. There were developed modeling software tools of the multilayer perceptron application for the cryptographically strict remote users' identification.

There is developed software for one of the reviewed scheme protocols by Java language, created abstract model of identification system's operation.

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ДП 4682.01.000 ВП	Відомість дипломного проекту	1	
3	А4	ДП 4682.02.000 ТЗ	Технічне завдання	4	
4	А4	ДП 4682.03.000 ПЗ	Пояснювальна записка	74	
5	А4	ДП 4682.04.000 Д1	Схема ресурсів системи	1	
6	А4	ДП 4682.05.000 Д2	Діаграма класів	1	
7	А4	ДП 4682.06.000 Д3	Схема роботи програми	1	
8	А3	ДП 4682.07.000 Д4	Схема алгоритму	1	
9	А4	ДП 4682.08.000 Д5	Лістинг реалізації ідентифікації	18	

Зм.	Арк.	№ докум.	Підпис	Дата	ДП 4682.01.000 ВП		
Розробивла		Можаровська С.А.			Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації Відомість дипломного проекту	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Марковський О.П.					
Н. контр.		Сімоненко В. П.				НТУУ “КПІ”, ФІОТ, каф. От, гр. ІО-63	
Затверд.							

Технічне завдання до дипломного проекту

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ 2

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ 2

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ 2

4. ДЖЕРЕЛА РОЗРОБКИ 2

5. ТЕХНІЧНІ ВИМОГИ 3

   5.1. Вимоги до продукту, що розробляється 3

   5.2. Вимоги до програмного забезпечення 3

   5.3. Вимоги до апаратного забезпечення 3

6. ЕТАПИ РОЗРОБКИ 4

1. НАЙМЕНУВАННЯ ТА ОБЛАСТЬ ЗАСТОСУВАННЯ

Технічне завдання поширюється на розробку нового методу використання нейромежових технологій для криптографічно строгої ідентифікації віддалених абонентів систем обробки інформації колективного доступу. Областю застосування – підсистеми контролю доступу інтегрованих систем зберігання та обробки даних.

2. ПІДСТАВИ ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи кваліфікаційно-освітнього рівня “бакалавр комп’ютерної інженерії”, затверджене кафедрою спеціалізованих комп’ютерних систем Національного технічного університету України “Київський політехнічний інститут імені Ігоря Сікорського”.

3. МЕТА ТА ПРИЗНАЧЕННЯ РОЗРОБКИ

Мета проекту полягає в підвищенні ефективності криптографічно сторогої ідентифікації віддалених користувачів інтегрованих систем зберігання та обробки інформації за рахунок дослідженні можливостей використання нейромережових технологій для ідентифікації на основі концепції нульових знань. Розробка призначена для підвищення рівня захищеності підсистеми контролю та розподілення доступу користувачів інтегрованих систем зберігання та обробки інформації.

4. ДЖЕРЕЛА РОЗРОБКИ

4.1 Захариудакис Лефтерис. Метод быстрой аутентификации удаленных пользователей на основе концепции “нулевых знаний” // Наукові записки Українського науково-дослідного інституту зв’язку. 2017.- № 1 (45).– С.109-117.

4.2. Kittichokenai K. Secret Key-based Identification and Authentication with a Privacy Constraint / K. Kittichokenai, G. Caire. // IEEE Trans. Inf. Theory.- Vol. 62.- 2016.- № 11.- P. 6189-6203.

4.3 Murukesh C. An Efficient Gait Recognition System Based on PCA and Multi-Layer Perceptron / C.Murukesh, K.Thanushkodi // Life Science Journal. – 2013.- Vol.10.- P.1024-1029.

5. ТЕХНІЧНІ ВИМОГИ

5.1. Вимоги до продукту, що розробляється

5.1.1 Для реалізації ідентифікації в рамках концепції нульових знань використовувати незворотне перетворення, що реалізується багатошаровим персептроном.

5.1.2 Метод формування сеансових ключів – навчання персептнопа для формування матриці вагових коефіцієнтів

5.1.3 Кількість сеансових ключів – не менше 300.

5.1.4 Ймовірність правильної ідентифікації користувача за його сеансових паролем – 0.999

5.1.5 Ймовірність хибної ідентифікації фейкового користувача – 0.004.

5.1.6 Час ідентифікації користувача – не більше 10 мкс.

5.2. Вимоги до програмного забезпечення

- Операційна система MS Windows 10
- Visual Studio 2017
- C++11

5.3. Вимоги до апаратного забезпечення

- Процесор рівня Intel i5 і вище.

– Оперативна пам’ять не менше 500 МБ.
– Вільне місце на жорсткому диску не менше 100 МБ.

6. ЕТАПИ РОЗРОБКИ

	Дата
Вивчення літератури	20.12.2019
Створення та узгодження технічного завдання	15.01.2020
Вивчення літературних джерел	27.01.2020
Розробка структури та алгоритмів ідентифікації	14.02.2020
Розробка програмної моделі	01.05.2020
Відлагодження програми та виправлення помилок	15.05.2020
Оформлення документації дипломного проекту	06.06.2020

Пояснювальна записка
до дипломного проєкту
на тему: «Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації»

Київ – 2020 року

ЗМІСТ

ВСТУП 3

РОЗДІЛ 1 5

АНАЛІЗ ПРОБЛЕМИ ЕФЕКТИВНОСТІ ІДЕНТИФІКАЦІЇ ВІДДАЛЕНИХ АБОНЕНТІВ В СУЧАСНИХ УМОВАХ. ОГЛЯД НИНІ ІСНУЮЧИХ МЕТОДІВ ІДЕНТИФІКАЦІЇ 5

1.1.Аналіз вимог до систем ідентифікації віддалених абонентів в сучасних умовах.....5

1.2.Огляд нині існуючих схем ідентифікації.....10

Висновки до розділу 1 25

РОЗДІЛ 2 26

РОЗРОБКА СПОСОБУ ВИКОРИСТАННЯ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ КРИПТОГРАФІЧНО СТРОГОЇ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ.....26

2.1.Дослідження варіантів використання методів ідентифікації, що базуються на криптографічно строгій концепції “нульових знань”.....26

2.2.Дослідження варіантів використання персептрону для ідентифікації на основі концепції “ нульових знань ” 33

2.3.Розробка моделі криптографічно строгої ідентифікації на основі багатошарового персептрону.....38

2.4.Теоретичне та експериментальне дослідження моделі.....43

Висновки до розділу 2 51

РОЗДІЛ 3 54

					ДП 4682.03.000 ПЗ		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробила		Можаровська С.А.			Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації Пояснювальна записка	Літ.	Аркуш 1 / Аркушів 74
Перевір.		Марковський О.П.					
Н. контр.		Сімоненко В. П.				НТУУ “КПІ”, ФІОТ, каф. От, гр. ІО-63	
Затверд.							

РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ІДЕНТИФІКАЦІЇ З ВИКОРИСТАННЯМ ПЕРСЕПТРОНУ.....54

3.1.Задачі, покладені в основу схеми ідентифікації…….54

3.2.Розробка програми.....56

3.3.Програмна реалізація протоколу ідентифікації.....59

3.4.Продуктивність.....63

3.5.Реалізація.....64

Висновки до розділу 3 68

ВИСНОВКИ.....69

СПИСОК ЛІТЕРАТУРИ.....72

ДОДАТКИ

ВСТУП

Стрімкий розвиток мережевих і телекомунікаційних технологій сприяє розширенню застосування розподілених систем і наданню дистанційного доступу широкому загалу користувачів до їх інформаційних та апаратних ресурсів. Це дає змогу підняти на значно вищий рівень можливості широкого загалу користувачів під час розв'язання їх наукових і прикладних задач, гарантуючи ефективний доступ до інтегрованих інформаційних ресурсів, котрі чималі за розміром об'ємів пам'яті та обчислювальних ресурсів. Крім того, це гарантує використання обчислювальних можливостей складних комп'ютерних систем високоефективно.

Одним із найважливіших факторів ефективності розподілених систем вважається зведення до нуля можливості несанкціонованого отримання доступу до їх інформаційних ресурсів та здійснення контролю за дотриманням легальним користувачем, а також окремих складових розподілених систем прав доступу. Ба більше, популярним напрямком розвитку стало не лише ускладнення та збільшення кількості компонентів розподілених систем та розширення галузі їх використання, а й лавиноподібне зростання кількості їх користувачів.

Описані чинники розкривають потребу здійснення вдосконалення засобів ідентифікації абонентів розподілених систем. Надто, коли їх використання поширюється у галузі людської діяльності, неконтрольоване втручання у котрі може спричинити тяжкі наслідки. Прикладами цього є система керування повітряним транспортом, система, що забезпечує безпеку життєдіяльності людини та інші. Поширення застосування розподілених систем в галузі економіки стимулює стрімке збільшення економічних злочинів, які здійснюються шляхом отримання доступу до секретної конфіденційної комерційної інформації. Отже, розширення сфер розподілених систем потребує збільшення ефективності систем ідентифікації

їх абонентів за допомогою зменшення ризиків неконтрольованого отримання доступу до їх обчислювальних і інформаційних ресурсів.

Властиве для нинішнього етапу розвитку розподілених систем стрімке збільшення кількості їх абонентів потребує адекватного підвищення продуктивності ідентифікації.

Позаяк підвищення надійності захисту від неконтрольованого отримання доступу потребує, зазвичай, зростання об’єму часових ресурсів для його здійснення, отже комплексне розв’язання проблеми зменшення ризиків неконтрольованого отримання доступу, а також збільшення продуктивності здійснення контролю за доступом можу бути здобуто тільки за рахунок розробки повністю нових методів та засобів ідентифікації абонентів.

Зважаючи на описане раніше, тема бакалаврської роботи є актуальною для нинішнього етапу розвитку інформаційних і комп’ютерних технологій.

РОЗДІЛ 1

АНАЛІЗ ПРОБЛЕМИ ЕФЕКТИВНОСТІ ІДЕНТИФІКАЦІЇ ВІДДАЛЕНИХ АБОНЕНТІВ В СУЧАСНИХ УМОВАХ. ОГЛЯД НИНІ ІСНУЮЧИХ МЕТОДІВ ІДЕНТИФІКАЦІЇ

1.1 Аналіз вимог до систем ідентифікації віддалених абонентів в сучасних умовах

Ідентифікація віддалених абонентів – є уособленням однієї з трьох опорних задач захисту інформації та забезпечує захист від надання незаконного доступу до даних та ресурсів багатокористувацьких систем [1].

Користувачу необхідно зареєструватися, щоб увійти в систему та розпочати роботу в ній. Ідентифікації – це процедура розпізнавання абонента при вході в систему, яка, як правило, реалізується за допомогою наперед визначеного імені (ідентифікатора) або інформацією про користувача, що також сприймається системою [2].

На сьогодні сервіси, які забезпечують безпеку, працюють в розподілених мережах, в яких потрібно враховувати імовірність появи загрози, як з мережевого середовища так само і з локального. Виділяють такі види загальних загроз:

1. Здійснення несанкціонованого фізичного доступу зловмисником до обчислювальної системи, в якій працює сервіс безпеки;
2. Спроби обходу зловмисником заходів захисту, які направленні на отримання незаконного доступу до системи;
3. Підміна користувача – процес, коли злочинець отримує доступ під виглядом уповноваженого користувача, а саме системного адміністратора. Здійснення цього можливо попередньо перехопивши дані для ідентифікації;

4. Підміна серверу – процес, під час якого злочинець видає свою систему за легальний сервер. Дана дія надає зловмисникові можливість маніпулювання користувачем, для отримання від нього конфіденційної інформації;
5. Несанкціонований доступ до конфіденційної інформації, зокрема і несанкціоноване маніпулювання даними, втручання в реалізацію їх шифрування або розшифрування;
6. Несанкціоноване змінення даних, зокрема тих, які захищені криптографічними методами.

Базова концепція доведення користувачем знання якоїсь інформації лежить в основі майже всіх існуючих систем ідентифікації. Взаємодія двох сторін: користувача і системи, які прийнято позначати А і В відповідно, розглядаються в структурному плані. Слід звернути увагу, що на практиці є множина $\Psi=\{A_1,A_2,\ldots,A_n\}$ користувачів, усім з них надаються легальні права для доступу до системи.

За такої умови слід брати до уваги такі критерії ефективності системи ідентифікації віддалених користувачів як: витривалість системи до спроб несанкціонованого проникнення незареєстрованих користувачів ($U\notin\Psi$) в систему та об’єм ресурсів, які затрачуються на здійснення процесу ідентифікації.

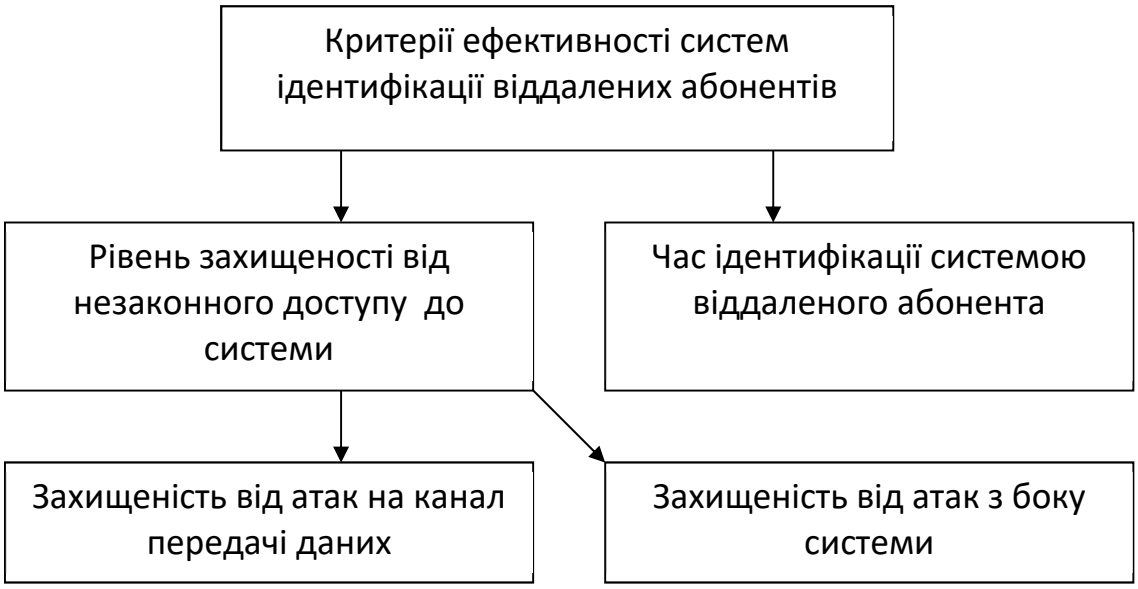


Рис.1.1 Критерії ефективності засобів ідентифікації віддалених абонентів

Само собою зрозуміло, що ефективнішою є система несанкціонований доступ до якої реалізований складніше та затрачено менше обчислювальних ресурсів на це. Під час визначенні ефективності ресурсам називають [3]:

- машинний час;
- об’єм пам’яті;
- час використання ліній комп’ютерної мережі.

Так як, всі відомі на сьогодні системи ідентифікації віддалених користувачів базуються на знанні ними певної інформації, а також ідентифікації – це процес впевненості в тому, що користувач, який намагається отримати доступ до системи, володіє цією інформацією, тому діяльності систем ідентифікації віддалених користувачів необхідно включати дві наступні стадії:

1. Стадія ініціалізації – під час якої абонент генерує або отримує секретну інформацію $V_{A,}$ яка в подальшому буде гарантувати йому доступ у систему.

2. Стадія перевірки легальності доступу – це стадія під час якої система перевіряє, що даний абонент дійсно знає інформацію V_A.

Забезпечення неможливості несанкціонованому користувачу практичного доступу в систему ідентифікації віддалених абонентів інтегрованих систем обробки інформації є головною вимогою до даної системи. Для висунення більш деталізованих вимог до архітектури систем ідентифікації слід дослідити головні методи несанкціонованого проникнення до інтегрованої системи, що має обмежений доступ. Згадана раніше система для обробки інформації спроможна обслуговувати n абонентів, що формують множину Ψ, для кожного *i*-го з яких, $i=1,..,n$ встановлена своя інформація I_i, яка потрібна для подальшої ідентифікації, а також, інформація D_i, що визначає чи надавати право доступу до ресурсів інтегрованої системи. Будь-який з користувачів для забезпечення отримання доступу до своїх ресурсів D_i, повинен володіти ключовою інформацією C_i.

Незаконне отримання абонентом, який не належить до легальних користувачів (множина Ψ), доступ до ресурсів інтегрованої системи, або отримання прав для доступу *k*-тим легальним користувачем $A_h \in \Psi$ до даних і ресурсів, які є власністю іншого абонента D_j , $i \neq k$ – це все є метою розкриття.

Для несанкціонованого проникнення в систему неідентифікований користувач, який є абонентом комп'ютерної мережі, може :

- Надавати системі інформацію I_k для ідентифікації, щоб здійснити спробу входження в неї; причому описані замахи можуть здійснюватися не одноразова, число яких обмежено величиною *C*, верхню границю якої в будь-який момент можна визначити, але точне значення *C* залежить від багатьох факторів та визначається конкретною ситуацією;

- Здійснювати прослуховування ліній передачі ідентифікаційної інформації між інтегрованою системою та легальними користувачами;

- Перехоплення фрагментів інформації для ідентифікації, яку використовують для обміну легальний віддалений користувач і система під час ідентифікації та в подальшому здійснювати підміну частини інформації;

- Одержувати з певними обмеженнями право доступу до ідентифікаційної інформації легальних абонентів, яка довгий час зберігається в пам’яті системи.

Звідси випливає, що для реалізації будь-якої з наведених вище можливостей потрібно витрачати певні ресурси. Система має здійснювати надійний захист від незаконного проникнення до даних та ресурсів системи в будь-якій описаних можливостей зловмисника. Система для ідентифікації віддаленого користувача обов’язково має бути захищеною від будь-яких намагань здійснити незаконне проникнення в неї за допомогою застосування пароля або його закодованого представлення, одержаного через перехоплення інформації на лінії, яка застосовувалася під час легальної ініціалізації користувачем А. Цього можна досягнути за допомогою часових штампів [4], які вставляються в пароль та шифруються приватним закриваючим ключем, а також за допомогою застосування інтерактивних багатоходових протоколів, в яких передбачають застосування випадково згенерованих системою елементів. Ясно, якщо велика кількість кроків багатоходової взаємодії, то при спробі розкриття ймовірність її повтору мізерна.

Отже, проаналізувавши ресурси доступ до яких є зі сторони зловмисника, можна для систем ідентифікації віддалених користувачів висунути такі вимоги:

- під час передачі інформації для ідентифікації від легального користувача до системи від сеансу до сеансу вона має обов’язково змінюватися для того, щоб не допускати прослуховування телекомунікаційних ліній передачі даних, за допомогою яких зловмисник може отримати доступ до інформаційних ресурсів;

- ідентифікуюча інформація, за допомогою якої користувач здійснює ідентифікацію, зобов’язана точно визначати об'єм інформаційних ресурсів, доступних абонентові та не передаватися по телекомунікаційних лініях зв’язку;

- інформація, яка дозволяє в повному обсязі реконструювати ідентифікаційну інформацію користувача не має зберігатися в системі;

- обов’язково мають використовуватися спеціальні технічні і алгоритмічні засоби для забезпечення захисту інформації, яка використовується для визначення легальності користувача та даних про надання прав доступу для кожного з абонентів, для забезпечення безпеки від несанкціонованого читання або редагування;

- виходячи з того, що система для ідентифікації віддалених користувачів є системою масового обслуговування, тому вона має здійснювати високу швидкість виконання процесу ідентифікації користувачів та мати здатність до розпаралелювання.

Досліджуючи ефективність схем ідентифікації віддалених абонентів інтегрованих систем постає вагома проблема здійснення схеми ідентифікації: так як потрібно враховувати, що опрацювання обчислень, котрі мають зв’язок з ідентифікацією, система реалізує на потужних комп’ютерах, що надає змогу не покладати на обчислювальну складність алгоритму обробки істотних обмежень. З іншого боку, обчислення, котрі пов’язані з процесом виконання протоколу ідентифікації, зі сторони віддаленого користувача можуть реалізовуватися на персональному комп’ютері з великою розрядністю процесора й водночас відносно простих 8-розрядних термінальних контролерах, та, навіть, на старт картах.

1.2 Огляд нині існуючих схем ідентифікації

Переважна більшість систем безпеки, які використовують нині, застосовують механізм ідентифікації в основі якої лежить схема ідентифікатор користувач/пароль. В основному, користувачі, котрі користуються системою з парольною ідентифікацією, використовують паролі, які є легкими для запам’ятовування, що надає зловмиснику

можливість легкого злому. Отже, ідентифікація, яка базується лише на паролі, не здатна забезпечити потрібний рівень захисту. Для вирішення цієї проблеми застосовують генерацію паролів, які складаються з обраних випадковим чином символів, котрі важко підібрати [5]. Недоліком є те, що користувачам важче буде запам’ятати пароль, а також це може викликати низку труднощі під час використанні двох і більше систем.

На кожен етап життєвого циклу обчислювальної системи мають виконуватися наступні вимоги: усі засоби захисту зобов’язані бути мають бути готові до відключення системи так і до несанкціонованого впливу; захист повинен здійснюватися постійно і працювати в будь-якому з режимів.

Щоб здійснювати безперервну роботу система повинна бути забезпеченою потрібним рівнем швидкодії засобів захисту. Для того щоб виконання функції захисту здійснювалося швидше, потрібно зменшити об’єми обчислювальних ресурсів, котрі застосовуються під час реалізації даного захисту. В той час, при підвищені швидкодії необхідно пам’ятати про підтримку належного рівня надійності системи безпеки. Наведенні вище проблеми є головними пов’язаними з засобами захисту але їх розв’язання є досить складним та важким завданням.

Аутентифікація, котра базується на введені користувачем реєстраційного імені та паролю, є достатньо зручною для користувачів[6]. Кожному абоненту присвоюється унікальне ім’я (ідентифікатор) з відповідним заданим паролем. Ідентифікація за допомогою паролю є достатньо простою, тому, на сьогодні, вона надзвичайно часто використовується. Здійснюється пошук у реєстрі введеного суб’єктом реєстраційного імені, а пароль порівнюється зі збереженим. Доступ у систему надається, якщо вони співпадають, в іншому разі забороняється. Нажаль, описана аутентифікація не ідеальна та має такі основні недоліки:

– Зловмисник має можливість дізнатися пароль абонента із файлу з паролями або з його резервної копії;

– Злочинець ховаючись під маскою адміністратора вимагає користувача надати свій пароль або поміняти на вказаний ним пароль (злочинець також може видавати себе за ідентифікованого користувача для того, щоб змінити пароль цього абонента за допомогою адміністратора);

– Злочинець здійснює спроби отримати доступ до системи використовуючи ім’я користувача жертви та підібрати потрібний пароль базуючись на отриманих особистих даних про неї;

– Також для визначення імен користувачів та їх паролів, злочинець може здійснювати аналіз мережевого трафіку від клієнта до сервера, який був ним перехоплений.

Від деяких недоліків парольної ідентифікації можна автоматично уникнути за допомогою системи керування паролями. Запам’ятовувати паролі під час використання цієї системи не потрібно, тому що їх генерація здійснюється автоматично, це дає змогу підвищити стійкість ідентифікації, яка досягається за рахунок використання складніших паролів. Дана система також надає можливість зловмиснику перехопити пароль, але так як він є одноразовим злочинець не отримає ніякої вигоди з цього.

Кожному користувачеві в системі відповідає унікальний ідентифікатор. Вхід до системи та отримання дозволу для доступу до її даних та ресурсів здійснюється тільки після успішного завершення ідентифікації.

Для наглядного прикладу, розглянемо типову схему ідентифікації, котра використовується у великій кількості систем але з деякими розбіжностями. Кожному користувачеві присвоюється індивідуальний об’єкт ідентифікації, який можна записати парою $\{ ID_i, C_i \}$ [7]:

– IDi – індивідуальний ідентифікатор, котрий надає змогу отримати з множини всіх зареєстрованих користувачі і-го абонента системи;

– Ci – інформація, яка може підтвердити автентичність і-го користувача системи.

Для підвищення рівня безпеки об’єкт ідентифікації знаходиться у системі в неявному вигляді, а саме зберігається об’єкт-еталон, котрий містить у захищеному форматі інформацію. Цей об’єкт складається з пари { ID_i, C_i }, де $K_i = \xi(ID_i, C_i)$. Функція ξ повинна бути незворотною, а саме унеможливлювати будь-які спроби підбору значення K_i і коду C_i . Деяке граничне значення P_0 має бути меншим за інтенсивність праці потрібної для знаходження C_i знаючи K_i. Нажаль, з деякою ймовірністю може відбутися хибна ідентифікація, тому що для будь-якої пари імовірний збіг відповідних значень[8].

Основні властивості протоколу ідентифікації UNIX у загальному вигляді такі :

1. Користувач зобов’язаний пред’являє особистий ідентифікатор ID;
2. Система керування доступом здійснює перевірку зареєстрованих ідентифікаторів для з’ясування існування ідентифікатора, який дорівнює даному. За умови успішного проходження перевірки суб’єкт, котрий іменується і-м суб’єктом доступу, здійснив ідентифікацію. В протилежному випадку в отриманні доступу буде відмовлено;
3. Система керування доступом відправляє запит для вводу абонентом його інформації для ідентифікації C_i;
4. Система управління доступом визначає значення $Z= \xi(ID_i, C_i)$;
5. Система управління доступом робить порівняння значення Z і K_i.

Суб’єкт доступу здійсним ідентифікацію за умови, якщо значення збігаються. Інакше доступ до системи не буде надано.

Ідентифікація користувачів і генерація ключів та їх подальший розподіл по каналу зв'язку об’єднуються в протоколі із застосуванням центра ключів. На рис. 1.2 зображено найпростішу конфігурацію, тобто протокол вміщує дві сторони, третьою складовою є ЦГРК (центр генерації та розподілу ключів), для зручності називається сервером G. Генерація, реєстрація і комунікація –

три складові протоколу. Під час генерації сервер G здійснює генерування числових значень параметрів системи, до яких входить і його приватний і публічний ключ. Під час реєстрації сервер G здійснює ідентифікацію абонентів, для всіх об'єктів виконує генерацію ключової інформації і / або даних для ідентифікації, а також утворює маркер безпеки, у котрому знаходяться потрібні системі константи і публічний ключ сервера G, якщо він потрібен. Під час комунікації виконується протокол ідентифікації ключового обміну, останнім етапом якого є утворення спільного сеансового ключа. В описаній схемі кожен абонент володіє лише одним ключем, який є спільний з ЦГРК -центром. Кожна операція керування ключами здійснюється через ЦГРК -центр.

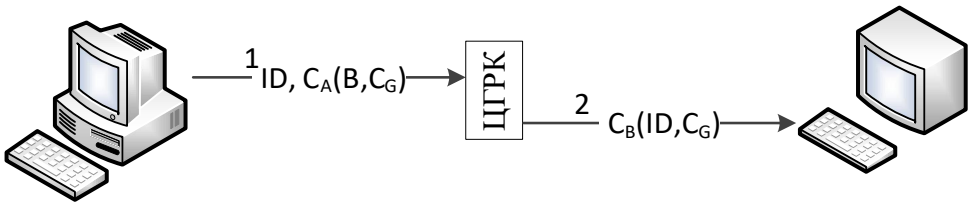


Рис. 1.2 Протокол ідентифікації з використанням ЦГРК – центра

Базовою для протоколу слугує така ідея: користувач обирає сеансовий ключ C_G, потім доводить до відому ЦГРК – центру, що він готується отримати доступ до сеансу використовуючи ключ C_G. Шифрування цього повідомлення здійснюється секретним ключем C_A, спільно володіють яким користувач і центр ключів [9]. Дешифрування повідомлення і отримання з нього ідентифікатора для системи та сеансового ключа здійснює центр розповсюдження ключів. Наступним етапом є утворення нового повідомлення ЦГРК – центром, котре складається з ідентифікатора особи абонент і сеансового ключа, потім відправляє його системі. Шифрування цього повідомлення здійснюється відповідно ключем C_B, який є спільним

приватним ключем ЦГРК – центра і системи. Дешифрувавши повідомлення, система має можливість дізнатися, що користувач намагається здійснити доступ до сеансу та ключ, який потрібно для цього застосувати. Надалі ідентифікація здійснюється само собою. ЦГРК - центр розуміє, що перше повідомлення надійшло від користувача, оскільки лише він міг здійснити шифрування повідомлення приватним ключем C_A. Точнісінько так, система розуміє, що друге повідомлення надійшло від ЦГРК, оскільки окрім системи лише центр володіє приватним ключ C_B.Незважаючи на позитивні сторони даного протоколу, присутні й недоліки – у нього відсутній захист від атаки повторного відтворення [10].

Інша група протоколів ідентифікації орієнтована на застосовуванні спільного приватного ключа. Розрізняють декілька видів. Розглянемо для прикладу схему ідентифікації користувачів, що базується на застосуванні спільного приватного ключа, може використовуватися також у протоколах типу вигук-відгук [11]. Основою для цього протоколу слугує принцип, що широко поширений в безлічі протоколів ідентифікації: одна сторона відсилає згенероване випадковим чином число іншій, яка в свою чергу виконує обробку цього числа спеціальною функцією, а потім повертає результат. Послідовність етапів протоколу представлено на рис. 1.3.

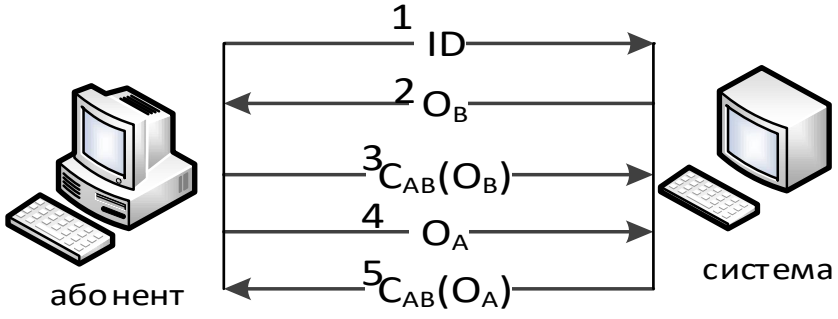


Рис. 1.2 Двостороння ідентифікація за допомогою протоколу вигук-відгук

1. Користувач надсилає системі особистий ідентифікатор;

2. Для підтвердження істинності користувача, система випадковим чином обирає велике число та відправляє користувачеві як вигук;
3. Використовуючи приватний ключ, котрий знають сторони, користувач здійснює шифрування цього повідомлення та надсилає результат системі;
4. Система обов'язково перевіряє повідомлення для того, щоб переконатися, що його шифрування здійснювалося за допомогою приватного ключа, який знає лише користувач;
5. Потім користувач має переконатися, що він спілкується із системою, а не з її підміною. Отже, наступні кроки протоколу симетричні: користувач надсилає вигук, а система відповідає на нього.

Кількість повідомлень описаного протоколу можна зменшити за допомогою об'єднання в одному повідомлені відповідь на попереднє та нового вигуку. Відповідна схема ідентифікації зображена на рис. 1.4 [12].

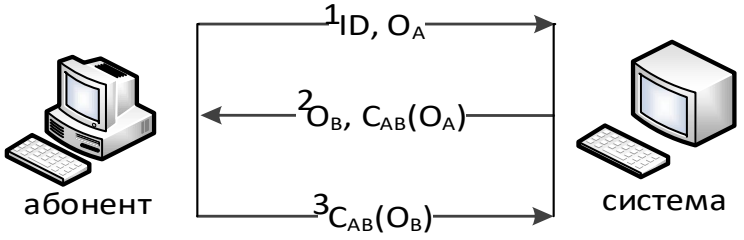


Рис. 1.4 Прискорений двосторонній протокол ідентифікації

Дана модифікація протоколу, одного боку, має виграш у час, але з іншого, надає можливість здійснення «дзеркальної атаки» (рис. 1.5) [13]. У такому разі, коли зловмисник матиме можливість одночасно здійснювати декілька сеансів зв'язку з системою, він зможе втрутитись в роботу протоколу. Видаючи себе за легального користувача зловмисник відправляє системі вигук. Система, зі свого боку, відповідає особистим вигуком. Далі зловмисник відриває другий сеанс повідомленням 3, та надсилає системі її власний вигук з 2-го повідомлення.

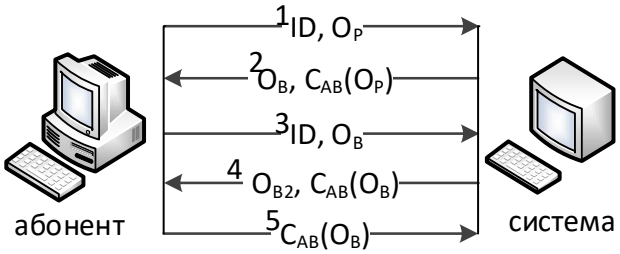


Рис. 1.5 Схема дзеркальної атаки

Система здійснює його шифрування та відправляє у вигляді повідомлення. Система шифрує його і надсилає назад повідомленням 4. Отже, використовуючи це зловмисник завершує перший сеанс та перериває другий.

Для того щоб уникнути цієї вразливості, у даній роботі [14], наведено чотири правила, що зобов'язаний виконувати протокол типу вигук-відгук:

1. Ініціатор сеансу під час кожного виклику зобов'язаний підтверджувати свою особу перед тим, як це здійснить інша сторона. У такому разі зловмисник не матиме можливості здобути цінну для нього інформацію перед тим, як здійснити підтвердження своєї особистості.
2. Різними ключами мають володіти ініціатор сеансу і той, що відповідає.
3. Кожна сторона обирає вигук різних непересічних наборів. Зокрема, ініціатор може використовувати парні числа, а інший – непарні.
4. Протокол повинен бути витривалим до такого роду атак, коли виконуються декілька паралельних сеансів та за допомогою одного вилучається інформація для іншого сеансу.

Протокол ідентифікації Нідхема-Шредера є складнішим. Даний протокол ідентифікації, в якому застосовується шифрування (симетричне або асиметричне) був висунутий Нідхемом і Шредером у 1978р. Протокол використовується під час двосторонньої ідентифікації із застосуванням виділеного сервера ідентифікації. Протокол Нідхема-Шредера, в якому

застосовується симетричне шифрування, описаний нижче. При використанні симетричного шифрування під час ідентифікації застосовується лише однин приватний ключ, який знають суб'єкти та сервер для ідентифікації AS. А також сервер ідентифікації знає два приватні ключі A і B, котрі не підлягають поширенню. Один з можливих варіантів протоколу показано на рис. 1.6.

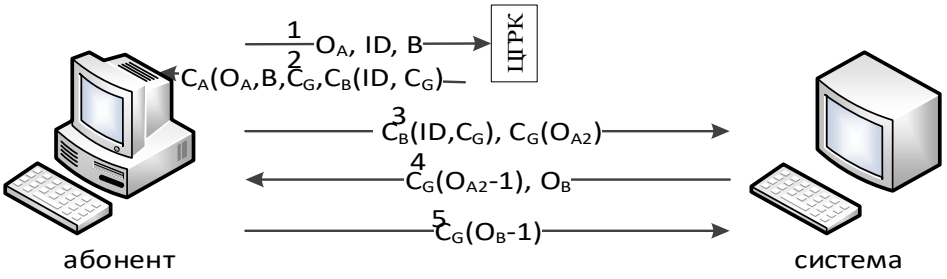


Рис. 1.6 Протокол ідентифікації Нідхема-Шредера

1. Для того щоб відкрити сеанс із системою, користувач відправляє центру повідомлення, котре складається із випадково згенерованого великого числа – ідентифікатор користувача і системи.
2. ЦГРК - центр надсилає у зворотному напрямку повідомлення 2, котре складається із згенерованого користувачем випадковим чином великого числа, сеансового ключа і квитка, який користувач відправляє системі. Ціль надсилання згенерованого випадковим чином великого числа у спробі переконати користувача в тому, що повідомлення 2 не повторно відтворене, а нове.
3. Користувач відправляє квиток, який містить сеансовий ключ зашифрований новим згенерованим випадковим чином великим числом.
4. На третє повідомлення система відповідає користувачеві відправляючи результат функціонального перетворення, для підтвердження власної автентичність.
5. Для цілковитого підтвердження, що система справді спілкується з користувачем, від відправляє повідомлення, котре складається із

зашифрованого сеансового ключа згенерованим системою випадковим чином великим числом.

Системи із використанням симетричного шифрування, котрі користуються лише одним ключем, потребують застосування безпечних каналів для передачі ключів. З іншого боку асиметричні системи з публічними та приватними ключами допускають передачу публічних ключів по незахищених каналах. Проте під час цього потрібно бути впевненим, що особа, з якою відбувається взаємодія з використанням алгоритмів з публічним ключем, є власником приватного ключа. Нажаль, наскільки б потужним не був захист протоколу, він все одно не буде захищеним від усіх можливих загроз. Даний протокол також не ідеальний. Для прикладу, якщо зловмисник якимось чином здобуде старий сеансовий ключ, то зуміє ініціювати навий запуск сеансу з системою, використовуючи отриманий ключ повторно відтворивши повідомлення 3 та вдавати легального користувача [15].

Вище було зазначено, що головними причинами застосування стандартизованих криптографічних алгоритмів для розв'язання задач ідентифікації віддалених абонентів є факт деякої гарантії забезпечення високого рівня криптостійкості під час реалізації ідентифікації криптографічних перетворень, ще й можливість істотно (на кільки порядків) збільшити швидкість ідентифікації за допомогою застосування апаратних засобів під час виконання стандартизованих алгоритмів, випуск котрих налагоджений серійно.

На практиці найчастіше [16] застосовуються стандартизовані криптографічні алгоритми з публічним ключем, адже вони реалізують максимально можливу концепцію монополії на ідентифікаційну інформацію.

Базою для майже всіх алгоритмів цього класу, на які нині існує великий попит, є складнорозв'язні задачі теорії чисел. Вкрай вжалим привілеєм несиметричних алгоритмів у зіставленні зі семеричними є вельми ширші можливості створення на їхній базі ефективних протоколів для захисту

інформації, котрі допускають можливість публічності одного з ключів (відкриваючого чи закриваючого), якщо інший ключ приватний. З'явившись наприкінці 70-х років [17] криптографія публічних ключів принесла з собою величезну кількість нових принципів організації захисту інформації, котрі чимала кількість дослідників [18] пов'язує з неймовірним стрибком "нової епохи розвитку криптографії".

На сьогодні, висунуто і підготовлено достатньо велику кількість несиметричних алгоритмів, серед яких алгоритм Ель Гамаля, Рабіна, RSA, алгоритми Мерклі-Хеллмана, ECC є одними з найбільш поширених і відомих.

Алгоритм Мерклі-Хеллмана вважається історично першим алгоритмом шифрування [19], у якому є публічний ключ, котрий базується на задачі виконання факторизації суми або задача "пакування рюкзака".

Процес дешифрування -- це відновлення даних компонент вектора інформаційного блоку використовуючи відомі значення V і компонент вектора Q. Припустимо, що для будь-якого q_j , j=1,...,n-1: $q_1+q_2+...+q_j< q_{j+1}$, то даний вектор ваг сильно зростаючий, якщо ж умова не здійснюється та $\exists\ q_k = \sum q_t$, $t\in\{1,2....,k-1\}$, тоді даний вектор нормальний. Зрозуміло, що задача дешифрування зі застосуванням сильно зростаючого вектора ваг буде простішою, матиме однозначний розв'язок та реалізуватиметься тривіальним чином, а під час застосування нормального вектора ваг характеризуватиметься неоднозначністю та виконанням складнорозв'язної задачі знаходження факторизації суми.

Зміст алгоритму Мерклі-Хеллмана лежить в тому, що деяким чином здійснюється формування двох векторів ваг: $Q_ш$ - публічний шифруючи, який є нормальним вектором та $Q_д$- приватний дешифруючи, що є сильно зростаючим. Вектори пов'язані один з одним за допомогою чисел a, b, с таким чином, що $q_{iш} = q_{iд}\cdot a \bmod b$. Процес шифрування повідомлення відбувається блок за блоком, а під час дешифрування здійснюється обрахунок $V'=V\cdot c \bmod$

b з наступною факторизацією суми S’ з використанням компонентів сильно зростаючого вектора $Q_д$. Обрані числа *a*, *b*, *c*, пов'язані один з одним деякими відношеннями та є приватними.

На практиці даний алгоритм застосовується при n>300, що забезпечує практично повне унеможливлення розв’язання задачі перебором. Втім існує можливість розкриття даного алгоритму використовуючи аналітичні методи за допустимий для практики час [20]. Розглянутий алгоритм вимагає найменших об’ємів обчислення під час зіставлення з іншими несиметричними алгоритмами.

Дві схеми описані нижче є найбільш характерними прикладами застосування несиметричних криптографічних алгоритмів з публічним ключем робота яких полягає в тому, що під час ініціалізації абонент А створює за допомогою відомих алгоритмів приватний ключ закриття SC_A та публічний ключ відкриття RC_A. Відповідно, система в пам’яті зберігає деяку множину ключів відкриття кожного легального зареєстрованого абонента системи, а також множину їх паролів (також можуть бути імена користувачів, адже описаний протокол не передбачає секретність паролів). Спочатку розглянемо схему, котра відноситься до категорії схем із використанням дворазового циклу ідентифікації, вона описує ідентифікацію віддаленого абонента таким чином:

1. Абонент А у відкритому вигляді надсилає системі особистий пароль R.
2. Система ініціює перевірку наявності отриманого паролю у списку паролів зареєстрованих абонентів, потім знаходить ключ відкриття користувача RC_A, здійснює формування рядка O випадковим чином та надсилає його віддаленому абонентові А.
3. Абонент А зашифровує одержаний рядок за допомогою всласного приватного закриваючого ключа SC_A та надсилає шифроване представлення повідомлення $M=F(O,SC_A)$ системі загального користування.

4. САВК за рахунок застосування публічного ключа відкривання здійснює дешифрування одержаного повідомлення М, за умови, якщо воно повністю співпадає зі згенерованим раніше повідомленням О, то абонент А отримує доступ до ресурсів системи.

Описана вище схема ідентифікації віддалених абонентів доволі ефективна та проста, однак одним з її недоліків є те, що нелегальний абонент отримує всю інформацію, котра потрібна для викриття несиметричного алгоритму шифрування, за допомогою відомих вхідного і вихідного кодів попри невідомий ключ закриття SC_A [21].

Наступна сема, по описаній вище класифікації, відноситься до групи схем зі застосуванням одноразової ідентифікації абонента, здійснюється за таким алгоритмом:

1. Віддалений абонент А утворює повідомлення, що складається з конкатенації пароля R та тимчасового штампу (TS), утім таким чином, щоб здобутий в результаті рядок по довжині дорівнює довжині інформаційного блока, який використовується несиметричним алгоритмом шифрування.

2. За допомогою приватного ключа закривання SCA абонент А зашифровує повідомлення: M=F(R||TS,SCA), потім надсилає кодом М за рахунок мережі в систему.

3. САВК санкціонує виконання перебору ключів відкриття абонентів зі списку зареєстрованих користувачів і за допомогою кожного зі обраних ключів реалізує розшифрування тексту М, обрання з дешифрованого блоку розрядів пароля, а також зіставлення його зі списком паролів легально зареєстрованих абонентів системи. За умови збігання паролів абонент А має право отримати доступ до виділеним ресурсам системи.

З описаного випливає, що в останній з розглянутих схем час здійснення ідентифікації набагато вищий (орієнтовно в *n* раз).

Сьогодні популярності набуває ідентифікація за допомогою фізичних об’єктів. Одним із найпоширеніших варіантів протоколів даного типу є

протокол під час якого користувач зобов’язаний підтвердити знання пароля, PIN чи особистий ідентифікатор (у залежності від способу ідентифікатора) та продемонструвати особливий предмет, який може бути представлений у вигляді смарт-карти, USB-токена та інших об’єктів. Зазвичай, суб’єкт крім підключення картки також повинен ввести пароль, це дає змогу уникнути застосовування картки зловмисниками.

Такі картки розділяють на дві групи: картки з нанесеною магнітною смужкою та картки з вбудованим чіпом (мікросхемою). Об’єм інформації на карті з магнітною смужкою, котрий зчитується терміналом та надсилається до центрального комп’ютера, становить приблизно 140 байт. Термінал може здійснити операцію ідентифікації користувача незважаючи на втрату з головним комп’ютером, це досягається за допомогою того, інформація на картці містить абонентський пароль [20]. Шифрування паролю здійснюється з використанням ключа, котрий знає виключно банк. Нажаль з’являється високий ризик застосування даних карток, адже обладнання потрібне для зчитування та запису інформації, котра міститься на карті, у відкритому доступі за достатньо низькою ціною.

Карти з вбудованою мікросхемою (чіпом) [21] поділяють на такі типи: карти зі збереженою сумою та смарт-карти. Місткість об’єму пам'яті карт зі збереженням суми невисока (частіше за все вона становить менше 1 Кбайта). Ця карта застосовує технологію збереження суми під час вилучення її зі зчитувального пристрою, а також під час вимкнені живлення [19]. Вносити зміни до суми, котра зберігається на карті, може виключно зовнішній центральний процесор, який знаходиться в зчитувальному пристрої, отже дані карти не містять центрального процесора.

Можна помітити тенденцію, що основний об’єм роботи по забезпеченню захисту беруть на себе смарт-картки [13]. Дані смарт-карти складаються з спеціального центрального процесора з розрядністю 8, тактовою частотою в 4 МГц, ПЗП 16 Кбайт, EEPROM 4 Кбайт, 512 байт непостійної оперативної

пам'яті та каналом зв’язку для здійснення обміну інформацією зі зчитувальним пристроєм. Смарт-карта представляє з себе крихітний але захищений від зовнішнього впливу комп’ютер, котрий зданий реалізовувати спілкування з центральним комп’ютером, для здійснення аутентифікації абонента [10]. Смарт-карти дозволяють застосовувати всілякі схеми ідентифікації. Проаналізуємо приклад, який базується на описаному раніше протоколі вигук-відгук. Сервер відправляє 512-розрядне число, згенероване випадковим чином, смарт-карті, котра в свою чергу добавляє до нього 512-розрядний пароль, який знаходиться в програмованому ПЗП. Наступним кроком є піднесення до квадрату одержаної суми та відправка середніх 512 бітів знову на сервер. Сервер здатний повторити зроблені дії, так як він знає пароль абонента, потім виконує порівняння з результатом.

Вадою всіх фіксованих криптографічних протоколів являється те, що пізніше його можна буде зламати зробивши непотрібною. Розв’язанням даного недоліку стало застосування інтерпретатора Java, до якого здійснюватиметься завантаження у виді двійкової Java-програми криптографічний протокол та виконання в режимі інтерпретації. Що забезпечує можливість завантаження нового протоколу, якщо минулий було зламано. Однак, розглянутий вище метод модифікації зумовлює більш повільну роботу смарт-карти.

Завдячуючи здатності визначити автентичність користувача без здійснення передачі конфіденційної інформації за допомогою каналів зв’язку, протоколи з нульовою передачею інформації [3] зайняли нішу систем, у яких високий рівень безпеки, для прикладу в картках електронного платежу.

Висновки до розділу 1.

В результаті проведених досліджень, котрі були направлені на здійснення аналізу нинішнього стану проблеми ефективної ідентифікації віддалених абонентів розподілених систем обробки інформації, розгляду наявних протоколів та методів ідентифікації можна виділити наступні висновки:

1. Головним недоліком ідентифікації, яка базується на паролях є недостатній рівень захисту від спроб здійснення несанкціонованого доступу, різноманіття типів якого в теперішніх умовах різко зростає. Описаний недолік визначений відсутністю в системах ідентифікації, які базуються на паролях, розвинених криптографічних механізмів, які надають можливість сеансової модифікації ідентифікуючої інформації, вразливість від отримання несанкціонованого доступу з боку системи даних, які застосовуються під час ідентифікації, недотриманням одного з головних принципів захисту інформації – присутності лише одного єдиного власника секретних даних.

2. Ідентифікація віддаленого абонента, котра базується на концепції “нульових знань”, є найбільш надійною з теоретичного боку.

РОЗДІЛ 2

РОЗРОБКА СПОСОБУ ВИКОРИСТАННЯ ТЕХНОЛОГІЙ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ КРИПТОГРАФІЧНО СТРОГОЇ ІДЕНТИФІКАЦІЇ КОРИСТУВАЧІВ

В умовах, коли інформаційна інтеграція стала одним із найвпливовішим чинником прогресу в усіх галузях людської діяльності, технології захисту даних, розподілення доступу до них, забезпечення їх автентичності та цілісності розвиваються випереджаючими темпами.

В останні роки розвиток засобів захисту інформації відбувається в напрямку ускладнення алгоритмів, збільшення розрядності чисел і це, відповідно, має наслідком помітне зростання обчислювальної складності реалізації цих, по суті, допоміжних, в роботі комп'ютерних систем функцій обробки інформації. Найбільш критично ця проблема стоїть для методів ідентифікації користувачів розподілених систем. Надійний захист прав доступу може бути досягнутий лише в рамках використання криптографічно строгих методів ідентифікації, суттєвого перешкодою широкому застосуванню яких є висока обчислювальна складність.

2.1 Дослідження варіантів використання методів ідентифікації, що базуються на криптографічно строгій концепції "нульових знань".

Спеціальні схеми, котрі використовуються для аутентифікації розділяють на два типи: які застосовують паролі ("слабка" ідентифікація) та які базуються на "нульових знань" ("сувора" ідентифікація). "Слабкою" вважається схема з ідентифікацією, котра базується на паролі, через те, що під час її застосування не дотримується принцип лише одного володаря секретних даних.

Переважна більшість протоколів ідентифікації віддалених користувачів базується на концепції “нульових знань”. Сенс даної концепції проявляється в тому, що для того, довести власну автентичність, користувач повинен неявним чином продемонструвати знання деякої інформації, доступ до якої система не має, але здатна перевірити факт її наявності у користувача.

Система не містить жодної секретної інформації, котра надала б змогу відновити ідентифікаційні дані користувача, цей дало назву концепції "нульових знань”. Вагомим є те, що під час кожного запиту до системи користувачем відбувається генерація повністю нової ідентифікуючої інформації.

Отже, у теоретичному аспекті концепція “нульових знань” більше за інші відповідає описаним раніше правилам для системи ідентифікації користувачів. Дана концепція застосовує теоретично незворотні криптографічні перетворення. Суть цього в тому, що є алгоритм перетворення в прямому напрямку, проте безкомпромісно неможливо знайти аналітично алгоритм для перетворення назад.

Необхідний практичний рівень надійності ідентифікації для сучасності та близького майбутнього можуть забезпечити виключно методи, котрі засновані на концепції “нульових знань”.

У даній концепції застосовуються незворотні математичні перетворення. Нутро концепції в тому, що є алгоритм перетворення в прямому напрямку, проте безкомпромісно неможливо знайти аналітично алгоритм для перетворення назад. Велика кількість схем ідентифікації, котрі базуються на концепції “нульових знань”, щоб реалізувати дане перетворення застосовують аналітично нерозв’язувані задачі теорії чисел, для прикладу задача дискретного логарифмування.

Максимального рівня поширення під час практичного використання досягли такі методи: FESIS, Guillou-Quisquater, а також Schnorr.

Розглянемо метод FESIS. Суб'єкт обирає прості числа r і u, потім рахує їх модуль $m=r \cdot u$. Щоб згенерувати приватний та публічний ключі суб'єкт обирає число q, яке дорівнює квадратичному залишку по модулю m. По іншому, абонент обирає таке значення q, котре має такий x, що $x^2 \bmod m = q$ та q^{-1}, що $q \cdot q^{-1} \bmod m = 1$. Знаходиться найменше v, при якому $v^2 \bmod m = q^{-1}$. Модуль m та число q формують публічний ключ, у свою чергу число v – приватний ключ.

Під час реєстрації абонент надсилає системі власний публічний ключ: модуль m і число q.

У циклі ідентифікації абонент обирає випадковим чином число p і знаходить значення $x = p^2 \bmod m$, потім знайдене значення x надсилає до системи. Система ініціює реалізацію t циклів акредитації, усі з них здійснюють такі дії:

1. Система відправляє випадковий біт b абоненту.

2. Якщо b=0, то абонент надсилає в систему число p, і іншому випадку (b=1) абонент виконує розрахунок $y = p \cdot v \bmod m$ застосовуючи приватний ключ s та надсилає його системі.

3.За умови якщо b=0, то система виконує перевірку $x = p^2 \bmod m$, у протилежному випадку (b=1) здійснює перевірку $x = y^2 \cdot q \bmod m$, пересвідчуючись, що користувач знає $v = \sqrt{q^{-1}}$.

Зловмисник, котрий намагається незаконно отримати доступ до системи, видаючи себе за легального абонента не знає компоненти приватного ключа: число q та модуль m абонента. Здійснивши перехоплення k циклів ідентифікації зловмисник одержить показники з k_1 циклів при b=0 і k_2 циклів при b=1, $k=k_1+k_2$. Отже, зловмисник володіє сукупністю пар чисел $\langle p_i, x_i \rangle$, $\forall i \in \{1,2,\ldots,k_1\}: x_i = p_i^2 \bmod m$ $\langle p_j, y_j \rangle$, та $\forall j \in \{1,2,\ldots,k_2\}: y_j = p_j \cdot v \bmod m$. Сукупність пар $\langle p_i, x_i \rangle$ може застосовуватися для виконання підбору модуля m. Сукупність пар $\langle p_j, y_j \rangle$ може використовуватися під час спроби підбору приватного ключа v. Описані задачі з математичного боку рівноцінні

розкладанню числа на два співмножника, а також враховуючи розрядність більшу за 1024 вимагають ресурсів, котрі є недоцільними.

За умови, що зловмисник перебуває в системі, тобто йому відомий публічний ключ, а саме число q та модуль m суб'єкта, то він має можливість обрати будь-яке s та вирахувати $\zeta = s^2 \bmod m$; відправити системі ζ як x. У випадку отримання зловмисником біт запиту b=1, він надсилає системі число s. Система здійснює перевірку, що $\zeta = s^2 \bmod m$. Однак, у випадку отримання зловмисником від системи біт запиту b=1, він повинен надіслати системі результат обрахунку $y = s \cdot v \bmod m$ використовуючи приватний ключ v. Зрозуміло, що не знаючи приватного ключа v, зловмисник матиме можливості знайти значення v. Спроба підбору зловмисником приватного ключа s з математичної сторони дорівнює задачі пошуку значення q^{-1} знаючи q. Дана задача може бути вирішена за умови, якщо зловмисник володіє складовими співмножниками r і u, котрі утворюють $m=r \cdot u$. Однак, система не володіє даними співмножниками, це означає, що успішне виконання задачі підбору сеансового паролю зловмисником, який перебуває в системі, майже неможливо здійснити.

Існують варіанти [14] описаної схеми ідентифікації, складові послідовності $q_1,q_2,\ldots,q_u$ публічних ключів утворюються, як хеш-сигнатури конкатенації рядка I інформаційного описувача абонента зі спеціально обраними кодами $s_1,s_2,\ldots,s_u$. Елементи публічного ключа утворюються як: $q_1=X(I||s_1)$, $q_2=X(I||s_2)$, $\ldots,q_u=X(I||s_u)$, а обрання $s_1,s_2,\ldots,s_u$ відбувається так, щоб кожний елемент послідовності $q_1,q_2,\ldots,q_u$ був еквівалентним квадратичному від'ємнику mod n.

Розглянута схема стала базовою для цілої низки ідентифікаційних схем. Так, в Японії застосовується одна з модифікацій описаної схеми модифікація описаної схеми [15].

Головними недоліками схеми ідентифікації FFSIS на практиці є потреба в декількох циклах акредитації, котра зі свого боку має необхідність у в

відповідності кількості сеансів передачі даних. Затрачений для виконання сеансу передачі даних між користувачем та системою, час істотно залежить від трафіку в комп'ютерній мережі та набагато швидше часу затраченого на здійснення системою обрахунків, які мають зв'язок з ідентифікацією абонента. Здійснення декількох недовгих сеансів передачі даних під час ідентифікації кожного абонента значною мірою позначається на пропускній здатності мережевого інтерфейсу систем типу колективного доступу. Окрім цього, факт присутності декількох прогнозованих сеансів передачі даних із визначеним абонентам надає змогу зловмисникам сильно зашкодити проведенню успішного циклу ідентифікації. Отже, потреба в декількох сеансах передачі даних істотним чином гальмує процес ідентифікації абонента.

Не лише метод FFSIS та його модифікації [16] широко поширені в системах колективного, а й Guillou-Quisquater, котрий теж використовує концепцію строгої ідентифікації "нульових знань". Відповідно до нього, абонент обирає публічний пароль ς. Публічний ключ також складається з модуля *m*, який утворюється абонентом як добуток $m = r \cdot u$ простих чисел *r* і *u*, котрі тримаються в секреті. Абонентом вибираються такі числа *q* і *B*, що $(\varsigma \cdot B^q) \bmod m = 1$. Під час реєстрації абонент надсилає системі публічний пароль ς та модуль *m*.

Послідовність дій циклу ідентифікації:

1. Абонент генерує випадкове число *p*.

2. Абонент вираховує сеансовий пароль у вигляді: $R = pq \bmod m$ і надсилає обрахований код R до системи.

3. Система, одержавши код R випадковим чином формує число w, котре виконує умову $0<w< m-1$; число w система надсилає абонентові.

4. Абонент рахує $S = p \cdot Bw \bmod m$ і відправляє вирахуваний код S системі.

5. Система здійснює обрахунки $U=Sq \cdot \varsigma w \bmod m$. Абонент рахується легальним, якщо здійснюється умова U=R.

Метод використовує еквівалентність перетворень як математичну базу:

$U = S^q \cdot \varsigma^w \bmod m = (p \cdot B^w)^q \cdot \varsigma^w \bmod m = p^q \cdot B^{w \cdot q} \cdot \varsigma^w \bmod m = p^q \cdot (\varsigma \cdot B^q)^w \bmod m = p^q \bmod m = R.$ (2.1)

Головним недоліком описаного методу в нинішніх умовах є застосовування декількох сеансів обміну даними між абонентом і системою, а також потреба реалізація операції модулярного експоненціювання абонентом безпосередньо під час процесу ідентифікації. Що призводить до істотного збільшення часу ідентифікації абонента.

Метод Schnorr [16] теж потребує здійснення вибору абонентом простих чисел r та u так, щоб u повинен бути подільником r-1. Обирається число а так, що $a^u \bmod r = 1$. Генерується випадкове число g менше за u : $g < u$, яке і є приватним ключем абонента. Потім абонентом рахується публічний ключ у вигляді: $= a^{-g} \bmod r$, який надсилається системі у процесі реєстрації.

Послідовність дій циклу ідентифікації абонента:

1. Абонент формує випадкове число p : $p<u$, потім рахує число

   $x = a^p \bmod r$.

2. Абонент надсилає код x у систему.

3. Система утворює випадкове k-розрядне число l і надсилає його абонентові.

4. Абонент рахує $y = (p + q \cdot l) \bmod u$ та надсилає одержане значення в систему.

5. Система визначає $\Sigma = a^y \cdot q^l \bmod r$ і порівнює одержане значення з отриманим кодом x: якщо $\Sigma = x$, абоненту надаються права доступу.

Особливістю описаного методу ідентифікації, що використовує концепцію “нульових знань” є застосовування групи чисел менших за u порівняно невеликої розрядності, це дає змогу істотно зменшити обчислювальну складність операцій, котрі мають зв’язок з процесом

ідентифікації. Однак, порівняно невелика розрядність приватного ключа *s* меншого за *u* : *g* <*u* надає можливість для його підбору зловмисником.

Істотним недоліком методу є те, що цикл ідентифікації віддаленого абонента вимагає трьох сеансів обміну даними між абонентом та системою, що ваговитим чином гальмує процес ідентифікації.

З описаного слідує, що основою обчислювальної операцією великої кількості схем строгої ідентифікації віддалених користувачів є модулярне експоненціювання над числами, довжина котрих істотно перевищує розрядність процесору. Нинішні тенденції стійкого зростання розрядності чисел, складність обчислення виконання вказаного типу операцій зростають експоненційно, обганяючи темпи збільшення продуктивності комп'ютерних систем.

Випущено [16] результати спроб застосовування при виконані концепції "нульових знань" в якості неповоротних булевих функціональних перетворень. Саме це рішення дає змогу пришвидшити процедуру ідентифікації віддалених користувачів. Недоліками даного рішення є істотне обмеження кількості сеансів ідентифікації, складніша для реалізації процедура утворення незворотного та багатозначного булевого перетворення, великий об'єм пам'яті потрібний для зберігання в системі таблиць з даними про перетворення для кожного користувача. Істотним також є момент, що не здійснені об'ємні та всебічні дослідження рівня захищеності таких методів. Отож, даний метод так і не дійшов до практичного використання в нині існуючих та діючих протоколах ідентифікації користувачів.

Як зазначалося вище, основним недоліком, що перешкоджає широкому використанню теоретично "суворої" технології ідентифікації на основі "нульових знань" є висока обчислювальна складність використовуваних для її реалізації необоротних і неоднозначних перетворень на основі мультиплікативних операцій модулярної арифметики. Застосування

нелінійних булевих перетворень виключає ефективне застосування математичних методів для порушення функцій захисту, тобто забезпечує її "строгість" в теоретичному сенсі.

При цьому їх використання дозволяє істотно підвищити швидкість ідентифікації за рахунок того, що обчислювальна складність реалізації булевих функцій на порядки менше в порівнянні з модулярними операціями модулярної арифметики. Однак, як зазначалося вище, використання булевих перетворень має ряд недоліків, найважливішими з яких є великий обсяг відкритого ключа абонента і обмежене число циклів ідентифікації, складність генерації відкритого ключа доступу.

Тому, використання булевих функціональних перетворень спеціальних класів не виключає доцільність пошуку альтернативних способів реалізації базової концепції "нульових знань". Одним з напрямків такого пошуку може бути застосування перетворень, що мають властивість неоднозначності, але не забезпечують суворої математичної незворотності.

Проблема може вирішена лише розробкою нових підході, здатних радикально прискорити обчислювальну реалізацію криптографічно строгої ідентифікації.

2.2. Дослідження варіантів використання персептрону для ідентифікації на основі концепції “ нульових знань ”

Використання технологій штучного інтелекту може розглядатися як можливість розв’язання цієї актуальної сьогодні та вагомої для практики наукової задачі. Задача розпізнавання образів за деякими властивостями є класичною для штучного інтелекту [12]. Задача розпізнавання образів подібна морфологічно задачі ідентифікації віддалених користувачів [13]. Поширенню використання технологій штучного інтелекту для реалізації схем криптографічно строгої ідентифікації віддалених абонентів сприяла розробка

та налагодження серійного випуску мікросхем для персептронів та нейронних мереж, котрі стали дешевшими та ефективнішими [14].

Отож метою досліджень, котрі сконцентровані в даному розділі є дослідження та опрацювання можливого способу використання персептрону під час реалізації ідентифікації віддалених користувачів з використанням передової концепції “нульових знань” та розробка потенціального способу його застосування, а також дослідження ефективності ідентифікації, у якій використовуються технології штучного інтелекту, експериментальним способом.

Згідно основному правилу організації захисту інформації, сутність якого полягає у тому, що ступінь захисту визначається за допомогою виконання нерівності – потенційні прибутки її порушення (несанкціоноване отримання прав доступу) повинні бути нижчими в порівнянні з вартістю ресурсів, котрі потрібно витратити для порушення захисту. Нажаль, достатньо велика кількість багатокористувацьких систем для яких використання концепції "нульових знань", котра базується на математично строгих необоротних перетворень, є недоцільним. Так як потенційні вигоди отримання несанкціонованого доступу в даних системах є порівняно малими, тому для реалізації концепції "нульових знань" достатньо буде застосувати не такі складні неоднозначні перетворення, котрим не властива суворо математична незворотність. Останнім часом, висунуто ряд схожих перетворень для використання у межах концепції "нульових знань". В обчислюваному плані вони є ефективнішими в порівнянні з мультиплікативними операціями модулярної арифметики, а також їхня незворотність базується на NP-повних задачах. Схеми ідентифікації користувачів, котрі базуються на завданні перестановки ядер (PKP- Permuted Kernels Problem) [15], аналізі симптомів SD (Sendrom Decoding) [16], застосовуючи обмежені лінійні рівняння CLE (Constrained Linear Equations)

[17] – це приклади альтернативних варіантів використання концепції "нульових знань".

Застосування нейромережевих технологій, що базуються на моделі персептрона, є альтернативним варіантом втілення спрощеної версії концепції "нульових знань".

На сьогоднішній день існує ряд математичних моделей базового компонента нейронної мережі - персептрона [18].

Стандартними операціями персептронів є навчання та безпосередньо розпізнавання. В відомих варіантах [19] застосування персептрону для ідентифікації віддалених користувачів використовувався один його образ, який задавався певним набором числових даних. Користувач, до виконання реєстрації в системі вибирав пароль, який є певним набором даних (символьних або числових). після цього користувач здійснював навчання персептрону, тобто адаптивне налаштування його матриці вагових коефіцієнтів на цей набір даних. При цьому, в процесі навчання вводилися інші набори числових даних і вони задавалися як помилковими.

Таким чином, в процесі навчання персептрону користувач налаштовував його на надійне розпізнавання одного заданого паролю. Матриця налаштувань відсилалась в систему, де зберігалась в пам'яті за адресою, яка визначалася ідентифікаційним кодом звернення користувача до системи.

При виконанні циклу ідентифікації користувач посилав в систему пароль у вигляді згаданого вище набору даних. Цей пароль розпізнавався системою і таким чином виконувалася ідентифікація користувача, якому надавалося право доступу до ресурсів системи. При заданні інших даних персептрон з налаштуваннями користувача класифікував ці данні як сторонню особу.

Головна перевага описаного рішення полягає в реалізації захисту від атак з боку системи. Це означає, що зловмисник через систему не може дізнатися прямо пароль користувача і отримати доступ до ресурсів системи від його імені. Зловмисник в даному контексті може виступати у вигляді вірусної

програми, яка запущена в систему, або не доброчесного співробітника системи. В теоретичному плані такий захист ґрунтується на незворотності перетворень, які реалізує персептрон. Проблема незворотності функціонування персептрону не є в достатній мірі вивченою на сьогоднішній день [20]. Зокрема важно оцінити витрати ресурсів для визначення набору вхідних даних для правильної ідентифікації користувача при відомих значення вагових коефіцієнтів персептрону, які зберігаються в системі. В роботі [21] зазначається, що в принципі задача підбору паролю може бути зведена до розв’язання системи лінійних рівнянь, проте зазначається, що попередні дослідження показали, що такі рівняння мають багато вирішень і не можуть бути однозначно просто вирішені.

Очевидним недоліком відомих схем застосування персептрону для задач ідентифікації віддалених користувачів є незмінність паролю, або можливість його зміни в дуже обмежених границях. Це не забезпечує захисту паролю в процесі його передачі в систему. Іншими словами, зловмисник може отримати доступ до паролю шляхом пасивного прослуховування лінії передачі даних.

Передумовою створення систем криптографічно строгої ідентифікації на основі моделі персептрону є те, що всі ці моделі мають властивість неоднозначності. Це означає, що один і той же вихідний сигнал персептрона може бути викликаний великим числом варіантів значень вхідних сигналів. Дана властивість надає можливості використання персептрона в реалізації простішої версії ідеї “нульових знань”. Завдання пошуку зворотного перетворення, а саме отримання набору значень вхідних сигналів, які можуть викликати заданий вихідний сигнал, для персептрону є задачею NP-складності [22]. Так як у моделі персептрону реалізовано обмежено лінійний характер перетворень, завдання пошуку зворотного перетворення не є, у суворому математичному визначенні, аналітично нерозв'язною. Втім, якщо кількість вихідних сигналів велика, її розв’язок вимагає чималих

обчислювальних ресурсів, ціна котрих для широкого загалу багатокористувацьких систем в значній кількості перевищує вартісне співвідношення зиску від несанкціонованого доступу до затрачених ресурсів.

Матрицею M, яка складається з r рядків та c стовпчиків і елементи якої належать множині{-1;1}, можна описати найпростіший персептрон. Припустимо, що існує множина Z, яка містить r невід'ємних цілих чисел, кожне з яких не перевищує c. Задача пошуку n-компонентного вектора K, кожен з елементів якого знаходиться в межах від -1 до 1, такого, щоб його добуток з матрицею M дорівнював вектору $V = \{v_1, v_2, \dots, v_r\}$, $\forall i \in \{1,2,\dots,m\}: s_i \geq 0: M \cdot K = V$, і є задачею NP-складності.

При використанні персептрону в якості засобу криптографічно строгої ідентифікації паролі мають змінюватися при кожному зверненні користувача до системи. Для цього в процесі навчання персептрону користувач має навчити його розпізнавати різні набори числових даних, як єдиний образ користувача. Відповідно предметом досліджень є визначення надійності роботи персептрону по розпізнаванню образу користувача при різних вхідних паролях. Важливо отримати залежність між надійністю ідентифікації та кількістю можливих паролів. Цілком очевидно, що чим більша кількість паролів буде використовуватися, тим меншою буде ймовірність того, що користувач буде правильно ідентифікований. А ймовірність хибної ідентифікації, навпаки, з ростом кількості паролів буде зростати.

Зрозуміло, що ця залежність великою мірою залежатиме також від складності схеми персептрону і кількості його каскадів.

Це, в свою чергу, впливає на такі важливі для практики критерії ефективності, як вартість системи ідентифікації та його швидкодія. Відповідно, використання персептрону неефективне при програмній реалізації. Практичне застосування персептрону для задач криптографічно строгої ідентифікації можливе лише за умови застосування нейрочіпів. В якості прикладу можна навести надвелику інтегральну мікросхему MT19003

NISP фірми MicroCircuitEngineering. Ця мікросхема на апаратному рівні в рамках RISC-архітектури виконує сім команд управління багатошаровим персептроном. Наведена мікросхема є число цифровою. Для підвищення швидкодії широко застосовують аналого-цифрові мікросхеми персептронів, я яких частина операцій (додавання на множення) виконуються для забезпечення високої швидкодії на аналогових пристроях. В якості найбільш типового прикладу мікросхем персептрону такого класу можна навести Intel 8017NW ETANN (ElectricallyTrainableAnalogueNeuralNetworks). Ця мікросхема містить в собі 64 нейрона і здатна зберігати 10280 вагових коефіцієнтів [23].

2.3 Розробка моделі криптографічно строгої ідентифікації на основі багатошарового персептрону

Для досягнення поставленої цілі теоретично обґрунтовано, розроблено та досліджено модель криптографічно строгої ідентифікації віддалених користувачів з використанням багатошарового персептрону

На рис.2.1 зображена структура реєстрації користувача в системі колективного доступу, ідентифікація користувачів в якій здійснюється на основі використання персептрону.

Рис. 2.1. Структура реєстрації користувача в системі

Реєстрація нового користувача передбачає наступну послідовність дій:

1. Користувач створює *n* наборів ознак, кожен з яких ідентифікує користувача і слугує в якості сеансового паролю доступу в систему. В якості ознак можуть використовувати набори числових даних або даних, пов'язаних з моторикою рухів користувача (наприклад, цифрових характеристик координат точок "розпису" користувача мишею).

2. Здійснюється цикл навчання персептрону чи нейронної мережі шляхом завдання на входи багатошарового персептрону наборів вхідних параметрів, які асоціюються з сеансовими паролями користувача, а також випадкового згенерованих наборів вхідних параметрів, які асоціюються з невірними паролями.

Матриця M і вектор V застосовуються в ролі публічного ключа системи однакового для всіх користувачі. Кожну складову у системі випадковим чином генерують і надсилають користувачам під час реєстрації.

Користувач A після отримання від системи матриці *M* і вектора *V,* здійснює таку послідовність дій:

1.1. Спершу випадковим чином генерується *n*-компонентний вектор $K=\{k_1,k_2,\ldots,k_c\}$, $\forall i\in\{1,\ldots,c\}: k_i\in\{-1,1\}$.

1.2. Здійснюється множення згаданого вище вектора K на матрицю M. Результатом виконаного множення є вектор $Z=\{z_1,z_2,\ldots,z_r\}$:$Z=M\cdot K$, кожна з r компонент $z_1,z_2,\ldots,z_r$ якого є цілим числом, що не дорівнює нулю.

1.3. Здійснюється перетворення матриці M у модифіковану матрицю M'. Для виконання цього процесу послідовно аналізуються компоненти $z_1,z_2,\ldots,z_m$ вектора Z за умови, якщо i-та ($i\in\{1,\ldots,r\}$) компонента z_i вектора Z від'ємна ($z_i < 0$), то i-тий рядок шуканої модифікованої матриці M' формується як інверсія однойменного рядка матриці M: $\forall j\in\{1,\ldots,c\}$: $M'_{ij}=-1\cdot M_{ij}$. В інакшому випадку, тобто якщо, що i-та ($i\in\{1,\ldots,r\}$) компонента z_i вектора Z невід'ємна ($z_i \geq 0$), то i-тий рядок шуканої модифікованої матриці M' дорівнює однойменному рядку матриці M: $\forall j\in\{1,\ldots,n\}$: $M_{ij}= M'_{ij}$.

1.4. Обчислюється вектор V_A, який складається з r компонентів, як добутку модифікованої матриці M' на вектор K: $V_A = M'\cdot K$.

1.5. Код r-компонентного вектора V_A надсилається системі як публічний ключ користувача А. Ці коди, в порівнянні з M, відрізняються для будь-якого легального абонента системи.

Користувач А утворює випадковим чином наступні компоненти, що складають його приватний ключ:

- Вектор R формується для знаходження перестановок рядків матриці M і містить r неповторюваних компонентів $R=\{r_1,r_2,\ldots,r_r\}$, так що кожна j-та компонента вектора R не більша за r: $\forall j=1,\ldots,r$: $r_j\leq r$ та вказує на позицію i-того рядка в перетвореній матриці.
- Вектор $C=\{c_1,c_2,\ldots,c_c\}$, який використовується для опису перестановки стовпчиків матриці M включно зі зміною знаків їхніх компонент. Кожна i-та компонента вектора C знаходиться в інтервалі від $-c$ до c: $\forall i=1,\ldots,c$: $-c\leq c_i\leq c$ і вказує на позицію i-того стовпчика в перетвореній матриці. За умови, якщо c_i

<0, також знаки всіх компонентів *i*-того стовпчика матриці *M* додатково замінюються на протилежні.

- Випадковий вектор P.

Для наступних кроків до вектора перестановки стовпців $C = \{c_1,c_2,...,c_c\}$ формується зворотний вектор $C'=\{c_1',c_2',...,c_c\}$. Вектор C' реалізує зворотну перестановку стовпців матриці, включно зі зміною знаку: $\forall i \in \{1,...,c\}: c'_{c_i} = i \cdot sign(c_i)$.

При проходженні повного циклу запитів до системи користувач А здійснює таку послідовність дій:

2.1. Відповідно з обраними векторами *R* і *C*, які встановлюють правила перестановки рядків та стовпчиків, відбувається перестановка матриці *M*. За допомогою такої перестановки рядків і стовпчиків утворюється матриця *M'*: $M'=C(R(M))$.

2.2. Відповідно до вектора C^{-1} відбувається зворотна перестановка модифікованого вектор *K*. За допомогою такої зворотної перестановки буде сформовано модифікований вектор *K'* : $K'=C'(K)$.

2.3. За допомогою суми вектора *P* і модифікованого вектора *K'* обчислюється *c*-компонентний вектор *B*: $B= P + K'$

2.4. Вираховується хеш-згортки наступних кодів: спочатку конкатенації векторів перестановок *R* та *C*: $x_0 = X(R|C)$, потім окремо хеш-згортки векторів *P* та *B*: $x_1 = X(P)$ і $x_2 = X(B)$ і хеш-згортки добутків зазначених *c-компонентних* векторів на модифіковану матрицю *M'* : $x_3 = X(M'\cdot P)$ і $x_4 = X(M'\cdot B)$.

2.5. Коди x_1, x_2, x_3 та x_4 надсилаються до системи.

Система генерує випадковим чином ціле число $q \in \{0,1,2,3\}$ і надсилає дане число користувачеві А.

Якщо q=0, користувач А надсилає системі три коди: *R*, *C* та *P*. Система за допомогою порівняння $X(R|C)$ з отриманим раніше кодом хеш-згортки x_0

перевіряє правильність передачі кодів R й C; визначає хеш-згортку X(P) одержаного від користувача вектора P і зрівнює її з одержаним раніше x_1; також система здійснює перестановку рядків матриці M: R(M), виконує перестановку компонентів одержаного від користувача вектора P: C(P) і обчислює хеш-згортку добутку R(M)·C(P). За умови, якщо вона збігається з одержаним раніше кодом x_3, себто X(R(M)·C(P)) = x_3, то користувач А отримує права доступу.

Якщо q=1, користувач А теж надсилає системі коди трьох векторів: R, C та B. Система здійснює аналогічні обрахунки і перевірки з єдиною відмінністю, що те, що реалізовувалося для вектора P реалізується тепер для вектора B: за допомогою порівняння X($R|C$) з раніше отриманим кодом хеш-згортки x_0 перевіряється правильність передачі кодів R і C; визначається хеш-згортка X(B) одержаного від користувача c-компонентного вектора B і зрівнюється з одержним раніше x_2; здійснюється перестановка рядків матриці M: R(M), виконується перестановка компонентів одержаних від користувача вектора B: C(B), і обчислюється хеш-згортка добутку R(M)·C(B). За умови, якщо хеш-згортка добутку збігається з раніше одержаним кодом x_4, себто X(R(M)·C(B)) = x_4, то користувач А отримує права доступу.

Якщо q=2, користувач А виконує множення модифікованої матриці M' на вектор P і надсилає системі одержаний добуток – $M'·P$, потім аналогічним способом виконує множення модифікованої матриці M′ на вектор K_A', а одержаний добуток $M'·K_A'$ надсилає системі. Система, при одержанні зазначених кодів, визначає хеш-згортку добутку $M'·P$ і порівнює одержаний результат X($M'·P$) з раніше одержаним кодом x_3, після чого система визначає суму одержаних від користувача добутків $M'·P$ і $M'·K_A'$, потім утворюється хеш-згортка X$(M'·P+M'·K_A')$ даної суми, що порівнюється з одержаним від користувача кодом x_4. Врешті-решт, система звіряє, щоб усі компоненти добутку $M'·K_A'$ знаходилися в множині V_A. За умови, якщо всі вказані вище

умови здійснюються, то користувач А отримує права доступу до ресурсів системи.

Якщо q=3, користувач А надсилає системі сформовані вектори P й K_A'. Система, після одержання даних векторів визначає хеш-згортку X(P) вектора P і зрівнює її з кодом x_1, визначає хеш-згортку суми X($P+K_A'$) одержаних від користувача А векторів P і K_A', порівнює її з одержаним кодом x_4. За умови, якщо всі умови збігаються, то користувач А отримує права доступу до ресурсів системи.

За допомогою врахування особливостей виконання операцій під час ідентифікації обчислювальна складність описаної процедури ідентифікації може зменшитися.

2.4 Теоретичне та експериментальне дослідження моделі

Скалярне множення, що застосовується в згаданій раніше схемі ідентифікації, за основу якої взятий одношаровий персептрон, є найбільш масовою й трудомісткою операцією. Будь-який зі складових результату при виконанні такого множенні дорівнює сумі попарних добутків елементів матриць або векторів, що знаходяться в діапазоні від -1 до 1. Звідси випливає, що згадані раніше попарні добутки так само можуть знаходитися в такому ж діапазоні.

Твердження: За умови, якщо числа x та y знаходяться в діапазоні від -1 до 1: $x,y \in \{-1,1\}$, а також дані числа представлені в доповняльному коді, тому і добуток цих чисел може бути подане у вигляді: $x \cdot y = x \oplus y \vee 1$.

Як зазначалося вище, якщо множники x і y належать бінарній множині {-1,1}, тому результат обчислення добутку також належить даній множині. Водночас, якщо $x=y$, то $x \cdot y = 1$, а якщо $x \neq y$, то $x \cdot y = -1$. Розглянемо ці випадки. У першому випадку $x \oplus y = 0$ і тому $x \oplus y \vee 1 = 1$. У другому випадку для різних значень x і y можливо два варіанти. Спочатку припустимо, що x=1 та y=-1,

тоді $x \oplus y = 1 \oplus (-1) = -2$. Тому, $x \oplus y \vee 1 = -2 \vee 1 = -1$. Для обернених значень x та y результат буде співпадати.

Доведене вище твердження дозволяє виконувати скалярне множення набагато швидше, завдяки тому, що операцію множення компонент векторів виконувати не потрібно. Виявлене спрощення надає можливість прискорення процесу ідентифікації, оскільки, для забезпечення потрібного рівня захисту від застосування зловмисниками методу підбору, реально використовуються матриці розмірністю, що знаходиться в діапазоні від 200 до 300 [11].

Під час виконання прямого скалярного множення векторів, які складаються з c компонентів і ці компоненти знаходяться в неупакованому форматі, множення пари цих компонент потребує однієї операції множення, яка триває t_r. Час отримання результату скалярного добутку двох векторів дорівнює $c \cdot t_r + (c-1) \cdot t_x$, де t_y – час затрачений на виконання операції додавання. Будемо вважати, що тривалість t_r затраченого часу на операції множення в χ раз перевищує тривалість t_x затраченого часу на операції додавання ($\chi = t_r / t_x$), тоді час T_0 затрачений на виконання скалярного множення матриці, яка складається з r рядків й c стовпчиків, на вектор, який складається з c компонентів, визначається формулою:

$$T_0 = r \cdot (c \cdot t_r + (c-1) \cdot t_x) \approx r \cdot c \cdot (\chi + 1) \cdot t_x \qquad (2.2)$$

Під час використання описаного способу знаходження скалярного добутку варто зберігати вектор і матрицю, над якими виконується множення, в ”упакованому” форматі. Даний формат застосовує для зберігання кожного компонента 2 розряди, тому для зберігання вектора, який складається з c компонентів, буде затрачено $2 \cdot c/h$ машинних слів (кодів, які рівні розрядності процесора - h). Кожен компонент вектора x, над яким виконують операцію множенні, під час зберігання 2-х розрядів для доповняльного коду, значення $x = 1$ дорівнює коду пари розрядів 01, а $x = -1$ дорівнює коду 11. Під час застосування наведеного вище формату можливо одночасно здійснювати операцію множення над $h/2$ парами компонентів. Операція множення при

цьому буде зводитися лише з послідовного виконання порозрядної суми по модулю 2 і операції АБО з константою. А для операції додавання отриманих значень 2-бітових полів можливо буде застосувати табличний суматор або використовувати операції зсуву й додавання. Зважаючи на це і звертаючи уваги те, що час затрачений на виконання логічних операцій не перевищує час затрачений на виконання операції додавання, тривалість знаходження скалярного добутку двох векторів, котрі складаються з c компонентів, дорівнює $6 \cdot c \cdot t_x / h$. Тому час T_1 скалярного множення матриці, яка складається з r рядків й c стовпчиків, на вектор, який складаються з c компонентів, застосовуючи запропоновану вище організацію виконання операцій визначається формулою:

$$T_1 = \frac{6 \cdot r \cdot c \cdot t_x}{h} \tag{2.3}$$

Під час порівняння часу затраченого на виконання скалярного множення матриці на вектор для двох варіантів описаних раніше показує, що представлений спосіб виконання цієї операції може зменшити час її виконання в γ разів, у такому разі чисельне значення γ обчислюється наступним виразом:

$$\gamma = \frac{T_0}{T_1} = \frac{h \cdot (\chi + 1)}{6} \tag{2.4}$$

Розглянемо 32-розрядний процесор (h=32) і візьмемо $\chi = 3$, тоді час затрачений на реалізацію базової операції схеми ідентифікації, яка заснована на моделі персептрону, скалярного множення вектора на матрицю, під час використання представленого способу її виконання зменшується в 21 раз (γ=21.3) у порівнянні з традиційним способом її виконання. Також перевагою представленого способу реалізації базової операції є зменшення потрібного для зберігання операндів об'єму пам'яті орієнтовно в $h/2$ разів. У таку ж кількість зменшується час обчислення підсумовування векторів, а також виконання іншої операції – обчислення хеш-згортки.

Були проведені дослідження залежності стійкості представленої схеми ідентифікації користувачів з розрядністю параметрів r та c до спроб підбору кодів x, S_1 і S_2 для отримання несанкціонованого доступу до ресурсів системи, продемонстрували, що для реального використання розрядність вказаних раніше параметрів необхідно бути не менше 200.

Нижче вказано основні переваги застосування моделі персептрону для реалізації ідентифікації користувачів, в основі якої лежить спрощена концепція “нульових знань”:

1. Ідентифікація застосовує публічний ключ системи – матрицю M великого об'єму, яка є однаковою для всіх легальних користувачів. Публічні індивідуальні ключі користувачів – вектори V мають малий об'єм. Саме це забезпечує набагато менший обсяг пам'яті, який є необхідним для зберігання в системі інформації, яка у подальшому використовується для ідентифікації абонентів.

2. Генерація публічного ключа абонента відбувається істотно простіше та фактично не вимагає великих обчислювальних ресурсів.

3. Кількість сеансів звертання майже не обмежено.

Недоліками використання моделі персептрона для реалізації ідентифікації абонентів на основі спрощеної концепції "нульових знань" в порівнянні з застосуванням нелінійних необоротних булевих перетворень є:

1. Істотно менший рівень захищеності від несанкціонованого доступу до ресурсів системи, оскільки використовувані в моделі персептрона перетворення не є в строгому математичному сенсі незворотними.

2. Менша швидкість ідентифікації, що обумовлено тим, що обчислення булевого функціонального перетворення має на порядки меншу обчислювальну складність у порівнянні з операціями множення матриць.

3. Використання моделі персептрона, як і мультиплікативних модульних перетворень не виключає можливості повторного використання раніше застосованого легальним абонентом коду доступу до ресурсів системи.

Для практичної використання моделі персептрона при реалізації ідентифікації абонентів на основі спрощеної концепції "нульових знань" розроблений програмний продукт. Діаграма класів розробленого проекту показана на рис.2.2. В якості мови програмування для реалізації проекту використовувалася Java.

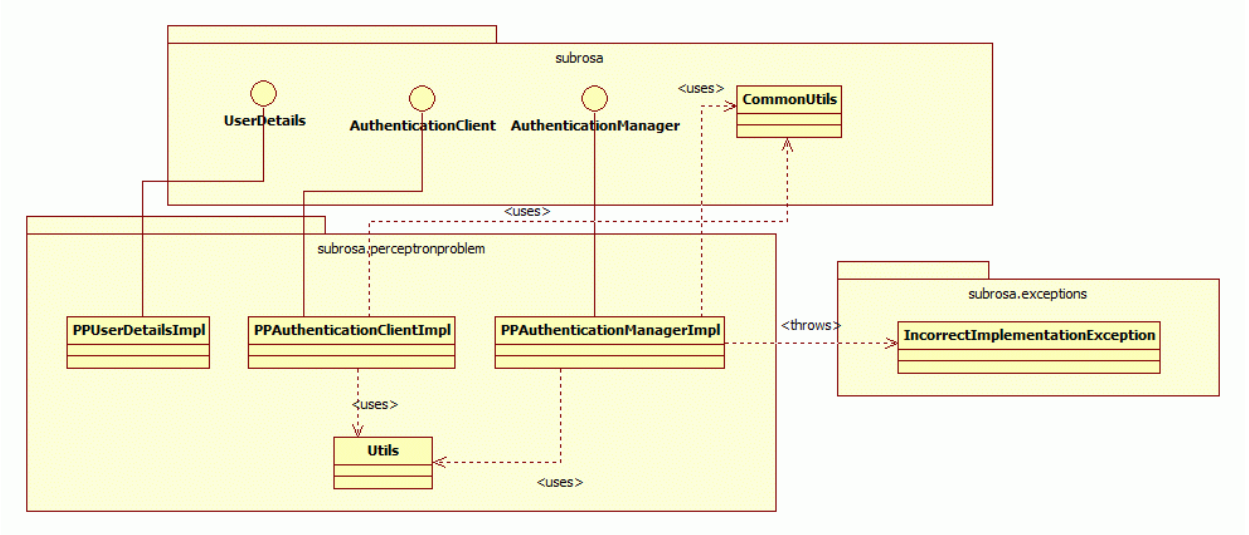


Рис.2.2. Діаграма класів програми ідентифікації з використанням моделі персептрона.

В рамках розробки спроектовані інтерфейси, які забезпечують верифікацію абонента.

Інтерфейс *UserDetails* містить методи *getPublicKey, getSecreteKey* , які повертають відкритий і закритий ключі абонента в тому форматі, в якому з ними працювати буде система. Робота системи по ідентифікації абонента реалізується в рамках інтерфейсу *AuthenticationClient*. При цьому, система повинна імплімінтувати інтерфейс *AuthenticationManager*. Останній містить методи для створення нового облікового запису абонента і виконання верифікації.

Пакет програми, крім зазначених вище інтерфейсів містить також клас утиліт *CommonUtils*, який інкапсулює в собі методи, які використовуються практично у всіх ідентифікаційних протоколах: генерація випадкової послідовності символів, алгебраїчні операції з векторами і матрицями. Формування випадкової послідовності символів здійснюється за допомогою вбудованого генератора мови Java: формується випадкове число в діапазоні 0...255, з подальшим вибором з цього числа ASCII-коду символу.

Операції з векторами і матрицями виконуються без застосування Java Collections Framework із використанням звичайних масивів зі скалярними типами даних для збільшення продуктивності.

Клас *PPUserDetailsImpl* інкапсулює в собі відкритий ключ, в вигляді матриці *W* і вектора значень *S,* а також секретний ключ абонента у вигляді вектора Y. Реалізація *AuthenticationClient,* клас *PPAuthenticationClientImpl*, використовує *PPUserDetailsImpl* і здійснює генерацію векторів перестановок *P* і *Q*. Для цього методи утилітного класу *Utils*.

При формуванні випадкової перестановки спочатку формується список їх можливих значень.

Потім циклічно генерується випадкове число, що належить інтервалу, який дорівнює довжині списку, яким індексується перестановка. Вона вставляється у вектор *P* або *Q* і виключається зі списку можливих перестановок. Цикл завершується, коли список стане порожнім.

Методи класу *PPAuthenticationClientImpl* формують матрицю *W'* і вектор Y, використовуючи перестановки *P* і *Q*. Клас *PPAuthenticationManagerImpl* імплементує інтерфейс *AuthenticationManager.* При верифікації абонента, в класі *PPAuthenticationManagerImpl* перевіряється імплементація абонента. У разі, якщо ця реалізація не відповідає описаному вище порядку (протоколу ідентифікації) генерується виняткова ситуація (за допомогою створення об'єкта класу *IncorrectImplementationException*).

Для дослідження залежності продуктивності ідентифікації від різних чинників був розроблений графічний інтерфейс. Вікно програми містить дві вкладки: для моделювання процесу реєстрації та для ідентифікації. Спочатку необхідно вибрати потрібну вкладку і заповнити параметри моделювання:

1. початкові значення розмірностей (*n* – число компонент векторів і кількість стовпців матриці, *m* – кількість рядків матриці);

2. кількість аналізованих значень розмірності (число точок на графіку);

3. кількість циклів моделювання для одного значення розмірностей;

4. крок з яким виконується збільшення розмірності.

Після натискання кнопки «Start» запускається потік, що виконує статистичне моделювання процесів реєстрації або ідентифікації абонента. Після закінчення реєстрації будується графік. Вид вікна з побудованим графіком показаний на рис.2.3.

На наведеному на рис.2.3. графіку представлена залежність часу формування відкритих і закритих ключів абонента при його реєстрації для двох варіантів їх реалізації - з використанням звичайної процедури скалярного добутку (верхня крива) і з застосуванням запропонованої вище модифікованої процедури (нижня крива). З наведених графіків видно, що при використанні модифікованої процедури скалярного добутку, заснованої на доведеній лемі, сумарний час реєстрації абонента зменшується, в середньому, на 35%, причому з ростом розмірності використовуваних векторів і матриць ефективність розробленої процедури в плані зменшення часу реєстрації збільшується. Відносно малий ефект в плані підвищення швидкості реєстрації запропонованої процедури обумовлений малою питомою вагою операцій скалярного добутку.

При ідентифікації абонентів ефект від використання запропонованої процедури скалярного добутку істотно збільшується.

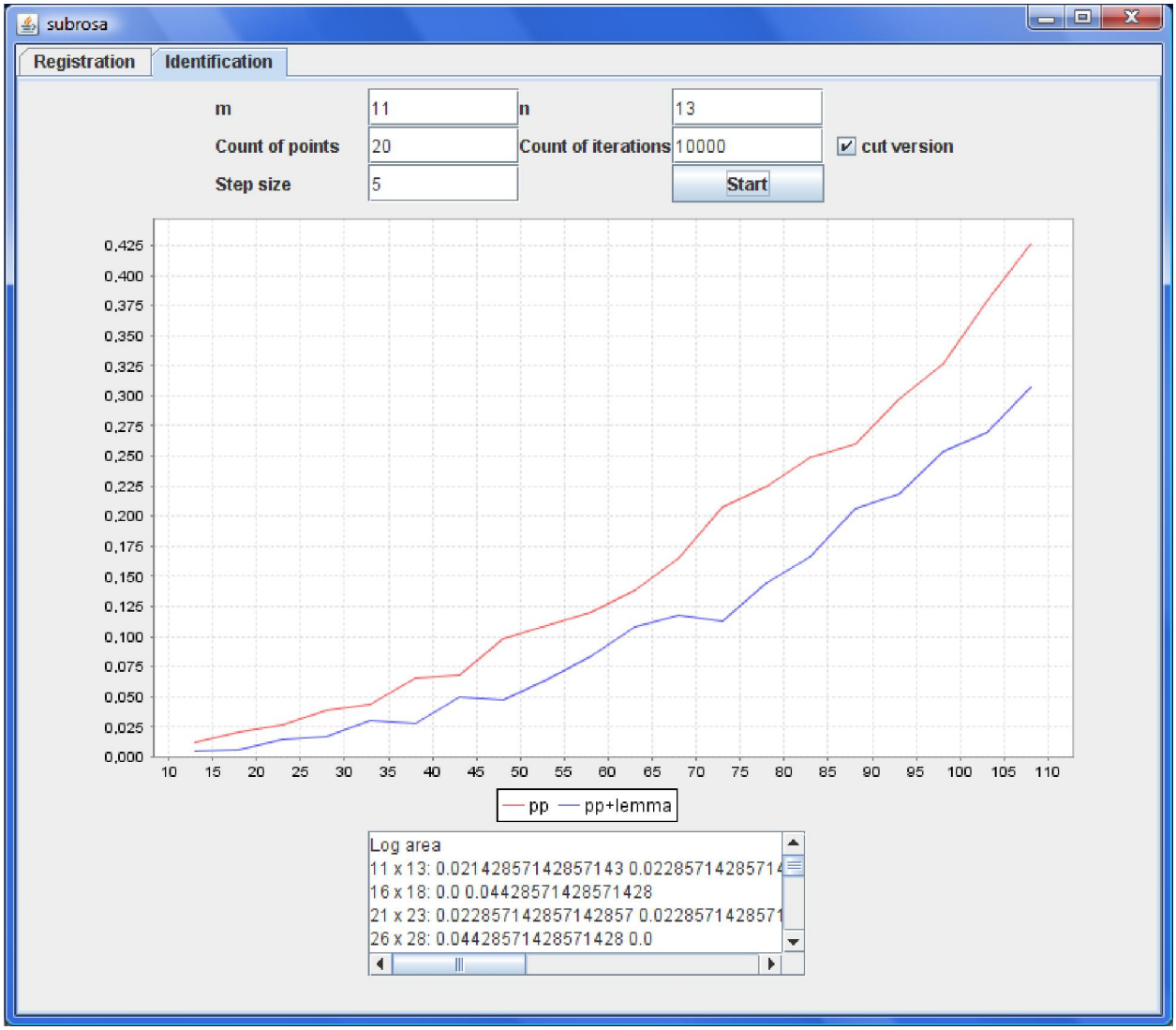


Рис.2.3. Експериментальні результати моделювання: залежності часу формування відкритих і секретних ключів від розмірності векторів і матриць, що описують модель персептрона.

Висновки до розділу 2

В результаті проведених досліджень, які складають другий розділ бакалаврської робити, і направлені на теоретичне обґрунтування, розробку та оцінку ефективності способу використання багатошарового персептрону для реалізації криптографічно строгої ідентифікації віддалених користувачів отримані наступні результати:

1. Методи, котрі реалізують концепцію “нульових знань” , математично базуються на аналітично нерозв’язній задачі дискретного логарифмування. З цього слідує, що основними операціями обчислення для методів є мультиплікативні операції модулярної арифметики. Щоб обчислювальна складність реалізації була меншою в існуючих методах реалізовані деякі спрощення, наприклад під час застосовування операції модулярного піднесення до квадрату використовуються числа з набагато меншою розрядністю. Подібні спрощення компенсуються застосуванням декількох (від 3-х до 20) сеансів для обміну даними між абонентом та системою, котрі істотно сповільнюють ідентифікацію абонентів. Отже, головний недолік нині існуючих методів реалізації теоретично строгої ідентифікації, котрі базуються на концепції “нульових знань” в тому, що вони мають високу обчислювальній складності їх опорних обчислювальних процедур. Зрозуміло, що швидкість ідентифікації обмежується великою обчислювальною складністю даних операцій та, відповідно, з'являється необхідність у зменшенні її.

2. Головним напрямком для підняття ефективності криптографічно строгої ідентифікації є підняття рівня швидкодії, котре можна досягнуте за допомогою: скорочення кількості циклів обміну ідентифікаційною інформацією між абонентом та системою, застосовування інакшого математичного базису незворотних перетворень так, щоб виконання обчислень мало б не таку велику обчислювальну складність і здійснювало прискорення обрахунків або полегшення схеми під час апаратної реалізації.

3. Показано, що багатошарові персептрони знаходять використання в криптографічному захисті інформації в якості умовно незворотних перетворювачів. Доведено, що рівень захищеності при цьому визначається об'ємом ресурсів, які потрібно витратити, щоб реконструювати по заданим ваговим коефіцієнтам персептрону вхідні сигнали, які формують на виході персептрона позитивний вихідний сигнал класифікації.

4. Показано, що ефективність використання персептрона для криптографічно строгої ідентифікації віддалених користувачів визначається здатністю персептрона вірно класифікувати різні набори вхідних даних, при цьому критеріями ефективності виступають кількість наборів вхідних даних, які класифікуються персептроном як правильні, ймовірність правильної класифікації та ймовірність хибної класифікації.

5. Використання багатошарового персептрону для реалізації криптографічно строгої ідентифікації віддалених користувачів з огляду на жорсткі вимоги до оперативності ідентифікації доцільно лише за умови апаратної реалізації персептрона на серійних цифрових та аналого-цифрових НВІМС або на програмованих матрицях.

6. Запропоновано метод застосування персептрона для криптографічно строгої ідентифікації віддалених користувачів, який включає процедуру навчання, реєстрації та циклу ідентифікації, який відрізняється вдосконаленою процедурою скалярного множення, за рахунок чого досягається прискорення процесу ідентифікації та також збільшення можливих сеансових паролів.

7. Теоретичне дослідження ефективності розробленого методу показало, що час затрачений на здійснення базової операції схеми ідентифікації, котра базується на моделі персептрона - скалярного множення матриці на вектор під час застосовування запропонованого способу її час затрачений на здійснення зменшується в 21 раз (β=21.3) у порівнянні з традиційним

способом її реалізації. Ще одна перевага розробленого запропонованого способу полягає в скороченні витрат ресурсів пам'яті приблизно на 40%.

8. Експериментальні дослідження запропонованого методу показали, що при тривалому навчанні та використанні багатошарових персептронів ймовірність правильної класифікації сеансового паролю користувача доволі висока і становить 0.98 при кількості сеансових паролів 200. При цьому ймовірність хибної класифікації випадкового пароля не перевищує 0.015.

РОЗДІЛ 3

РОЗРОБКА ПРОГРАМНИХ ЗАСОБІВ ІДЕНТИФІКАЦІЇ З ВИКОРИСТАННЯМ ПЕРСЕПТРОНУ

3.1 Задачі, покладені в основу схеми ідентифікації

Проблема персептрона **ПП** виникає не лише у вченні про персептрони Айсінга, а й у фізиці та штучному інтелекті машин, які самостійно навчаються.

Будемо вважати вектором (чи матрицею) вектор (чи матрицю), всі компоненти якого набувають значенням +1 чи –1.

Проблема персептрона **ПП**:

Дано : матрицю M, розмірністю $r \times c$.

Задача : знайти вектор B, розміром c, такий що $MB \geq 0$

Дана задача важка для розв'язання.

Можливо розробити для кожної NP-повної задачі протокол ідентифікації «нульових-знань» за умови, якщо задана одностороння хеш-функція. Але для пришвидшення і спрощення ми можемо розглянути лише один із випадків даної проблеми, а саме :

Зміщена проблема персептрона **ЗПП**:

Дано : матрицю M, розмірністю $r \times c$. Множина Z з невід'ємними цілими числами, розміром *r.*

Задача : знайти вектор B, розміром *c,* такий щоб

$$\{\{(MS)_i | i = \{1, \dots, r\}\}\} = Z. \quad (3.1)$$

*Я*к бачимо, розв'язок для **ЗПП** – це і є розв'язок для **ПП.** Тому зміщена проблема персептрона **ЗПП** має складніший розв'язок, ніж перша проблема персептрона **ПП**.

Пізніше безпечні значення *r* та *c* будуть уточнені з ефективністю атак. Доведемо, що ефективність атак для певних випадків залежить від кількості розв'язків.

Отже, для початку оцінимо середнє значення кількості розв'язків (M, Z) для зміщеної проблеми персептрона **ЗПП**, розміром $r \times c$, де обов'язково існує не менше одного розв'язку S.

- З одного боку, можна порахувати розв'язки асоційовані з випадком проблеми персептрона. Нехай HH(*r, c*) буде середньою кількістю розв'язків випадку проблеми персептрона з матрицею M розмірністю $r \times c$ (це велика формула).
- З іншого боку, потрібно оцінити ймовірність отримання заданої множини Z з елементами, що рівні добутку M на заданий розв'язок ПП: Hr,c,Z.

Для кожної множини Z ця ймовірність є меншою, ніж імовірність отримання множини E, елементи якої мають розподіл Гауса (тобто $|\mathrm{E}|_i = rh_{c,i}$, де $h_{c,i}$ це ймовірність для двох векторів розміром *c*, A і B, бути такими щоб $|A \cdot B| = i : h_{c,i} = 2^{-c+1}\left(\frac{c}{\frac{c+i}{2}}\right) \cong \sqrt{\frac{8}{\pi c}}e^{-\frac{i^2}{2c}}$, де c = i mod 2)

$$H_{r,c,Z} \leq r! \prod_{i=1}^{c} \frac{h_{c,i}^{rh_{c,i}}}{(rh_{c,i})!} \tag{3.2}$$

Тоді кількість розв'язків для середнього випадку з ЗПП менша ніж

$$HH(r,c) \times H_{r,c,\mathrm{E}}. \tag{3.3}$$

Проте більшості ефективних атак, буде необхідна найменша кількість розв'язків. Тоді оберемо *r* та *c* такими щоб

$$HH(r,c) \times H_{r,c,E} \leq 1. \tag{3.4}$$

В результаті бачимо, що для розмірів, які нас цікавлять (100<*r*<200), можемо наближено взяти *c=r+16*.

3.2. Розробка програми

Розпочнемо з генерації ключів. Для цілей криптографії використовуюся тільки випадки з відомими розв'язками. Також необхідно знати всі випадки, які мають хоча б один розв'язок. Щоб отримати такий варіант, спочатку випадково вибираємо вектор *S*, розміром *c*, який буде розв'язком майбутнього випадку і матриці *M*, розміром $r \times c$. Потім модифікуємо її наступним чином:

→ Якщо $(MS)_i < 0$, то заміняємо *i*-тий рядок M на його протилежне значення.

→ Якщо $(MS)_i \geq 0$, то не змінюємо цей рядок.

Тоді обчислюємо $Z = \{\{(MS)_i | i = 1, \dots, r\}\}$. В результаті (M, Z) – це випадки зміщеної проблеми персептрона, де V – розв'язок. До того ж цим методом усі "задовільні випадки" можуть з'явитися з ймовірністю, яка пропорційна кількості розв'язків у випадку **ПП**.

Також для цілей криптографії необхідно обмежити розмір чисел, що використовуються, з метою зберігати їх в сталій кількості біт. Таким чином будемо мати поле кінцевих розмірів з h елементів. Якщо обмежити складові добутку невід'ємними непарними цілими d (тобто $D \in \{1,3,\dots d\}^r$), то можна довести, що коли *c, h, d* такі що $2h > c + d$, ми маємо:

$$MS = D \Leftrightarrow MS = D \bmod h \qquad (3.5)$$

Було розглянуто кілька атак проти **ПП** і **ЗПП,** щоб оцінити безпечність можливого протоколу. Але, оскільки немає алгебраїчної структури даних проблем, жодна з маніпуляцій над матрицею не лишить її незмінною (маніпуляції на зразок розв'язку Гауса, які використовувалися в минулому проти PKP, CLE або будь-якої проблеми, основаної на корекційних кодах помилок, не допоможе тут). Отже, зрозуміло, що тільки (більш-менш інтелектуальний) вичерпний пошук чи вірогідна атака будуть успішні.

Перша атака проти **ЗПП** – це *головний вектор V*:

для всіх і $\begin{cases} V_i = +1, \text{ якщо } \{j | M_{\text{j,i}} = +1\} > \frac{c}{2} \\ V_i = +1 \text{ інакше} \end{cases}$ (3.6)

Теорема. Для всіх випадків $r \times c$, утвореного як показано нижче з розв'язком *S* та $r \leq c$, середня відстань Хемінга між *S* та *V* наближено рівна:

$$c \cdot \left(\frac{1}{2} - \frac{1}{\pi}\sqrt{\frac{r}{c}}\right) \approx 0{,}2c \quad (3.7)$$

Спершу, можемо змінити 20% елементів V, і розглянути добутки, але є $\binom{c}{0.2c}$ таких можливостей.

Тепер можна зафіксувати межу для c (і *r*), щоб прийняти звичайний робочий фактор $2^{64} \div c \geq 95$.

Щоб удосконалити атаку, можна застосувати ці зміни, починаючи з елементів V, чиї значення невизначені (тобто $\#\{\boldsymbol{j}|\boldsymbol{M_{j,i}} = +\mathbf{1}\}$ близько $\frac{\boldsymbol{c}}{\mathbf{2}}$). Але досить несподівано деякі з елементів, які здавалися досить задовільними, виявляються неправильними: близько 80% елементів мають бути відслідкованими. Тобто це вдосконалення не змінює межу *c*.

Через неефективність попередньої атаки, було розглянуто добре відомий імовірнісний алгоритм, який походить із штучного інтелекту, і відомий як симульована нормалізація. Ця атака намагається мінімізувати енергетичну функцію, визначену на кінцевому метричному просторі, вірогіднісним шляхом. Алгоритм симульованої нормалізації є удосконаленням алгоритмів спадного градієнту. Тоді як алгоритм спадного градієнту може збігатися в точку локального мінімуму і лишатися там, симульована нормалізація намагається залишити її через малу випадкову перестановку. Ці перестановки, які можуть бути важливими на початку, мають прямувати до нуля.

Такий алгоритм може бути ефективним тільки, якщо енергетична функція кусочно-неперервна (різниця між зображеннями двох сусідніх точок

обмежена малим числом). По цій причині симульована нормалізація не підходить для **ЗПП**, але вона має бути ідеальною для **ПП** з

$$E(S) = \frac{1}{2}\sum_{c=1}^{r}([(MS)_i] - (MS)_i) \quad (3.8)$$

цей алгоритм виявляється найефективнішим. Було проведено багато тестів на квадратних матрицях ($r=c$), і на деяких інших розмінностях, за один день було знайдено рішення лише для випадків **ПП,** чиї розмірності є меншими за 200.

Якщо було припущено, що кожен з розв'язків може з'являтися з однаковою ймовірністю, ми маємо повторювати ці атаки близько 0.7×# {розв'язки для **ПП**} разів, з метою знайти хороший розв'язок з імовірністю 0.5.

Потім можна обчислити робочий фактор таких атак з ймовірністю успіху рівною 0.5:

Таблиця 3.1 Робочий фактор атак

Розмірність	Кількість рішень для **ПП**	Час розв'язку (в секундах)	Час розв'язку Hn =1/2 (с.)	Робочий фактор* 2^n робочих операцій
101×117	$4.7 \cdot 10^9$	85	$399 \cdot 10^9$	64
121×137	$8.7 \cdot 10^{10}$	130	$11 \cdot 10^{12}$	68
151×167	$3.7 \cdot 10^{12}$	180	$666 \cdot 10^{12}$	74

*робочий фактор розрахований використовуючи процесори 60-70 MIPS

Далі можна дійти до висновку, що навіть малі розмірності достатньо безпечні. На додачу, якою б не була імовірнісна атака, вона не буде в змозі відрізнити задовільне рішення **ЗПП** від будь-якого рішення **ПП**. Тобто, якщо навіть припустити швидку атаку для **ПП** (*NP*-повна задача), яка потребує лише 1 секунду, щоб знайти розв'язок у випадку з розмірністю 141×157, робочий фактор буде більшим 2^{64}.

З цими результатами можна запропонувати *r = 101* та *c = 117* як безпечну розмірність задачі. Таким чином, для середнього випадку $|Z|_1 = 15, |Z|_3 = 14$, т. д.

$$\underset{\text{випадки}}{\text{Hn}}\ [\text{Max Z } > 33] < 0.3 \qquad (3.9)$$

Далі можна припустити, що не існує числа більше 33 (тобто *d=33*). Потрібно прийняти *h > 75*, і тому беремо *h = 127* (це оптимізує імовірність невдачі злочинця).

3.3 Програмна реалізація протоколу ідентифікації

Загальні дані: кілька цілих *h, c, d* таких, щоб $2h > c + d$ та хеш-функція X, що не допускає колізії.

Нехай *M* – матриця розмірністю $r \times c$, випадок **ПП** з розв'язком *S*. Нехай *Z* буде множиною з елементів *MS*.

Публічний ключ: *(M, Z)*.

Секретний ключ: *S*.

Той, хто доводить вибирає:

→ Випадкову передновку H над $\{0, \ldots, r-1\}$ (щоб переставити рядки M).

→ Випадкову знакову перестановку U над $\{0, \ldots, c-1\}$ (щоб переставити колонки M та помножити їх випадково на +1 або -1).

→ Випадковий вектор G з K_p^c.

1. Той, хто доводить, обчислює $M' = RMU, S' = U^{-1}S, L = G + S'$ та $x_0 = X(H|U), x_1 = X(G), x_2 = X(L), x_3 = X(M'G), x_4 = X(M'L)$, відправляє $(x_0, x_1, x_2, x_3, x_4)$ перевіряючому.

2. Перевіряючий випадковим чином вибирає $q \in \{0, \ldots, 3\}$ та відправляє тому, хто доводить.

3.

Той, хто доводить, відправляє:	Перевіряється:
$q = 0: (H, U, G)$	$x_0 = X(H\|U), x_1 = X(G), x_3 = X(HMUG).$
$q = 1: (H, U, L)$	$x_0 = X(H\|U), x_2 = X(L), x_4 = X(UMUL).$
$q = 2: (M'G, M'S')$	$x_3 = X(M'G), x_4 = X(M'G + M'S'),$ $= \{\{(M'S')_i\}\} = Z$
$q = 3: (G, S')$	$x_1 = X(G), x_2 = X(G + S'), S' \in \{-1, +1\}^c.$

Твердження. Протокол **3p zk** є інтерактивною системою доведення для З**ПП**. Нехай вірогіднісний суперник з поліноміальним часом приймається з імовірністю більшою за $\left(\frac{3}{4}\right)^n + e$ після n раундів, тоді існує вірогіднісна машина з поліноміальним часом, яка отримує секретний ключ Z із публічних даних або вихідних колізій для передавальної функції з переважною імовірністю.

Доведення. Розглянемо дерево $D(\varphi)$ для всіх 4^i виконань, що відповідають усім можливим рівнянням того, хто перевіряє, через і раундів, коли супротивник має фіксовану випадкову стрічку φ.

$$\chi = \underset{\varphi}{Hn}[D(\varphi) \text{має вершини з } 4 - \text{ма дочерними вітками}] \qquad (3.10)$$

Очевидно, що $\chi > e$, і через зміну супротивника $\frac{1}{e}$ раз він знаходить зі сталою ймовірністю дерево виконання с 4-ма дочірніми гілками. Повторюючи ще раз, ця імовірність може стати дуже близькою до представленої.

Дерево с 4-ма дочірніми гілками відповідає ситуації, коли 5 передач $(x_0, x_1, x_2, x_3, x_4)$ були проведені і коли супротивник може надати відповіді на 4 можливих запити перевіряючого.

Розглянемо відповіді:

$X(G_0) = x_1 = X(G_3)$ $\quad\quad$ $X(H_1MU_1G_1) = x_4 = X(B_2 + T_2)$

$X(L_1) = x_2 = X(G_3 + S_3')$

Поки не знайдені колізії для хеш-функції H, можемо розглядати:

$H = H_0 = H_1$	$L = L_1 = G_3 + S_3' = G + S'$
$U = U_0 = U_1$	$B = B_2 = H_0MU_0G_0 = HMUG$
$G = G_0 = G_3$	$B + T = B_2 + T_2 = R_1MU_1L_1 = HMUL$

такими, що $\{\{(M'S')_i\}\} = V$,$S' \in \{-1, +1\}^c$.

Отже, $B + T = HMUL = HMUG + T = HMUG + HMUL'$, де $T = HMUL'$.

Нехай $S = US'(S' \in \{-1, +1\}^c)$, тоді $T = HMS$, а значить $\{\{(MS)_i\}\} = Z$.

П’яти-кроковий ідентифікаційний протокол (**5p zk**)

1. Той, хто доводить, обчислює $M' = HMU, S' = U^{-1}S$та $x_0 = X(H|U), x_1 = X(G|S'), x_2 = X(M'G|M'S')$, відправляє (x_0, x_1, x_2) перевіряючому.

2. Перевіряючий випадковим чином обирає $i \in K_h^*$ та відправляє тому, хто доводить.

3. Той, хто доводить, обчислює $L = iG + S'$ та $x_3 = X(L), x_4 = X(M'L)$ і відправляє(x_3, x_4) перевіряючому.

4. Перевіряючий випадковим чином обирає $q \in \{0,1,2\}$ та відправляє тому, хто доводить.

5. Той, хто доводить, відправляє:	Перевіряється:
$q = 0: (H, U, L)$	$x_0 = X(H\|U), x_3 = X(L), x_4 = X(HMUL).$
$q = 1: (M'G, M'S')$	$x_2 = X(M'G\|M'S'), x_4 = X(iM'G + MS'), \{\{(M'S')_j\}\} = Z$
$q = 2: (G, S')$	$x_1 = X(G\|S'), x_1 = X(iG + S'), S' \in \{-1, +1\}^c.$

Властивості.

Теорема 7. Протокол **5p zk** є інтерактивною системою доведення для **ЗПП**.

Лема 8. Нехай вірогіднісний суперник з поліноміальним часом приймається з імовірністю більшою за $\left(\frac{2h-1}{3(h-1)}\right)^n + e$ після n раундів, тоді існує вірогіднісна машина с поліноміальним часом, яка отримує секретний ключ Z із публічних даних або вихідних колізій для передавальної функції з переважною імовірністю.

Використовуючи ідею переставної симуляції, у випадковій істинній моделі, можна показати, що обидва протоколи є «0-знаннями». Також необхідно застосувати статистичні незалежні властивості для хеш-функції.

Можливо розробити легкі версії (**3p light**, **5p light**) цих протоколів, які зменшують кількість необхідних кроків.

П'яти-кроковий ідентифікаційний протокол **(5p light)**. Ініціалізація та сама, що й в попередніх схемах.

1. Той, хто доводить, обчислює $M' = HMU, S' = U^{-1}S$та $x_0 = X(H|U), x_1 = X(G|M'G|S'|M'S')$, відправляє (x_0, x_1) перевіряючому.

2. Перевіряючий випадковим чином обирає $i \in D_h^*$ та відправляє тому, хто доводить.

3. Той, хто доводить, обчислює $L = iG + S'$ і $x_2 = X(L|M'L)$ та відправляє x перевіряючому.

4. Перевіряючий випадковим чином обирає $q \in \{0,1\}$ та відправляє тому, хто доводить.

5. Той, хто доводить, відправляє:

$q = 0: (H, U, L)$

$q = 1: (G, S', M'G, M'S')$

Перевіряється:

$x_0 = X(H|U), x_2 = X(L|HMUL).$

$\left\{\left\{(M'S')_j\right\}\right\} = Z, S' \in \{-1, +1\}^c$

$x_2 = X(A|B),$

$A = iG + S', B = iM'G + 2D + Y$

Ці легкі версії уже не є «0-знаннями». Однак інформація, що використовується, виявляється досить невеликою. Насправді при $q = 1$ перевіряючий дізнається про два вектора через їх відображення на M', тоді він може зробити деякі висновки про M', а далі і про H, U. Оскільки він знає вектор S', теоретично він знає частину бітів S. Але, оскільки перестановки H и U змінюються на кожному раунді, ця інформація не представляє ніякої користі.

3.4 Продуктивність

Продуктивність даної схеми подібна до вже існуючих лінійних:

Таблиця 3.2 Продуктивність схеми

	SD Stern	CLE Stern	PKP Shamir	ЗПП 3p zk	ЗПП 5p zk
Розмірність матриці	256×512	24×24	37×64	101×117	
Кінцівка поля	K_2	K_{16}	$K_{25\ 1}$	K_2	
Складність кращої відомої атаки	2^{68}	2^{52}	$>2^{100}$	2^{64}	
Кількість раундів	35	20	20	48	35
Публічний ключ (біт)	256	80	296	144	
Секретний ключ (біт)	512	80	384	117	
Біти, що передаються за раунд	954	824	832	896	1040
Загальна швидкість передачі (Кб)	4.08	2.01	2.03	5.25	4.44

Як бачимо з таблиці 3.2, з більш безпечним секретним ключем (робочий фактор більший 2^{64}) та з імовірністю успіху злочинця меншою 10^{-6} ідентифікація вимагає менше ніж 4.5 Кбайт передачі між тим, хто доводить, та перевіряючим (порівняйте з 2 Кб для PKP і CLE, та 4 Кб для SD). Це можна вдосконалити, використовуючи хеш-дерева.

Більше того, всі операції вкладаються в додавання та віднімання малих цілих (менше одного байта) по модулю 2. Вони добре застосовуються для мінімального оточення 8-бітних процесорів.

При використанні спільної матриці M, що зберігається в бітах генератора псевдовипадкових чисел, та при ключах:

- секретний ключ: випадковий вектор *S* розмірністю *c* (менше 15 байт)
- публічний ключ: вектор R такий, що $R_i(MS)_i \geq 0$, та множина $Z = \{\{R_i(MS)_i\}\}$ (18 байт).

Маємо зовсім невелику кількість байт порівняно з PKP, SD. Але варто не забувати, що для PKP, SD, CLE ця схема не основана на ідентичності. Це значить, що публічні ключі повинні бути певним чином підтверджені.

Вони вимагають лише простих операцій, тому програми дуже невеликі. Більш того, розмірність даних (загальних даних та ключів) зовсім невелика, і в результаті необхідна мала EEPROM.

Зберігається дуже мало тимчасових даних при обчисленнях, тому необхідна невелика RAM.

3.5 Реалізація

Один із запропонованих протоколів ідентифікації (**3p zk**) було реалізовано на практиці. Діаграму класів розробленого проекту представлено на рис. 3.1. У якості мови програмування для його реалізації було обрано Java.

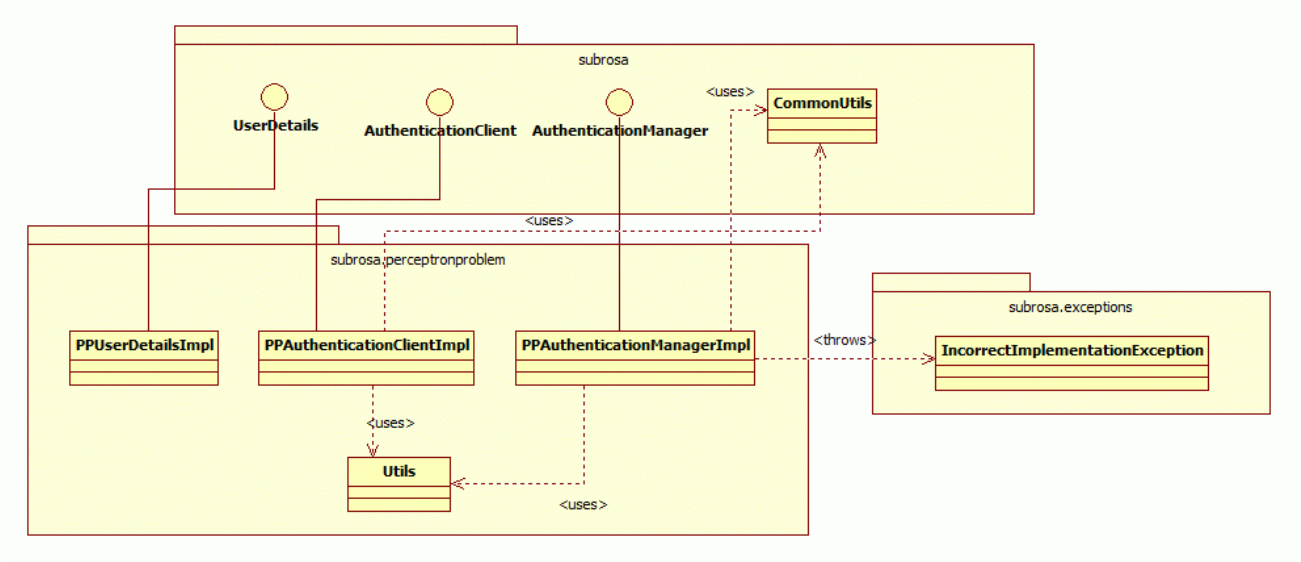


Рис. 3.1 Діаграма класів

Для цього було спроектовано інтерфейси, що забезпечують верифікацію користувача (пакет *subrosa*). Конкретна реалізація даних інтерфейсів, основана на проблемі персептрона знаходиться в пакеті *subrosa.perceptonproblem*.

Інтерфейс *UserDetails* містить методи *getPublicKey, getSecreteKey* , які повертають публічний та секретний ключі користувача у тому форматі, у якому з ними працюватиме клієнт (той, хто доводить). За пересилку даних до сторони, що проводить ідентифікацію користувача, відповідають реалізації інтерфейсу *AuthenticationClient*. Перевіряючи сторона має імплементувати інтерфейс *AuthenticationManager*. Він містить методи для створення нового облікового запису користувача та проведення верифікації.

Пакет *subrosa* окрім наведених вище інтерфейсів містить також утилітний клас *CommonUtils*, який інкапсулює в собі методи, що можуть бути корисними практично для всіх реалізацій ідентифікаційних протоколів: генерація випадкової послідовності символів, алгебраїчні операції з векторами та матрицями. Утворення випадкової послідовності символів відбувається за допомогою вбудованого в Java генератора: формується випадкове число в діапазоні 0...255, а потім забирається символ з ASCII-кодом рівним даному значенню.

Операції з векторами та матрицями відбувалися без застосування Java Collections Framework через звичайні масиви зі скалярними типами даних для збільшення продуктивності.

Клас *PPUserDetailsImpl* інкапсулює в собі публічний ключ, у вигляді матриці M та множини Z, і секретний ключ, у вигляді вектора S, розв'язку проблеми персептрона. Реалізація *AuthenticationClient,* клас *PPAuthenticationClientImpl*, використовує *PPUserDetailsImpl* та відповідає за генерацію перестановок H та U. Для цього використовуються методи утилітного класу *Utils*.

При формуванні випадкової перестановки спочатку утворюється список із можливих значень у перестановці.

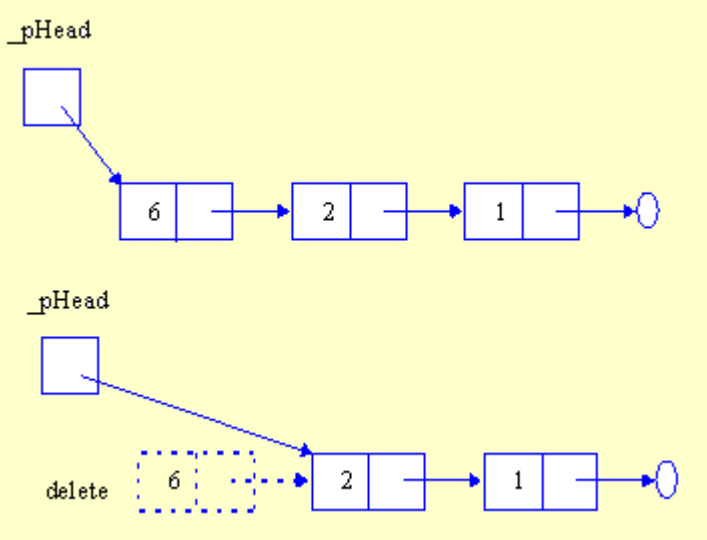


Рис 3.2 Формування випадкової перестановки

Далі, у циклі, генерується випадкове число у відрізку рівному довжині даного спису, елемент спуску, що знаходиться на місці, індекс якого рівний даному значенню, включається в перестановку та видаляється зі списку. Цикл завершується, коли список стане пустим.

Методи класу *PPAuthenticationClientImpl* обчислюють матрицю M' та вектор S', використовуючи перестановки H та U.

Клас *PPAuthenticationManagerImpl* імплементує інтерфейс *AuthenticationManager.* У його реалізації створення нового облікового запису користувачеві даються публічний та секретний ключі методом, описаним у розділі 3.2 Імплементація методу верифікації реалізує протокол **3p zk** описаний в розділі 3.3

Необхідно відзначити, що при верифікації у класі *PPAuthenticationManagerImpl* жорстко перевіряється імплементація клієнта. У випадку, якщо ця реалізація не є основаною на проблемі персептрона, генерується виняткова ситуація (через створення об’єкта класу *IncorrectImplementationException*).

Висновки до розділу 3

По результатам досліджень, які складають поточний розділ бакалаврської роботи, і направлені на створення засобів реалізації криптографічно строгої ідентифікації на основі використання багатошарових персептронів можна зробити такі висновки:

1.На основі аналізу задачі ідентифікації віддалених користувачів в рамках концепції нульових знань обґрунтовано вибір типу персептрона та організації програмних засобів моделювання його роботи.

2.Розроблені програмні засоби моделювання використання багатошарового персептрона для криптографічно строгої ідентифікації віддалених користувачів, в якому персептрона використовується в якості кластерного розпізнавача групи вхідних кодів, які асоціюються з сеансовими ключами користувача.

3.Було програмно реалізовано один із протоколів розглянутої схеми мовою Java, створено абстрактну модель функціонування системи ідентифікації.

4.Використання розроблених програмних засобів моделювання роботи багатошарового персептрона в якості незворотного перетворення для реалізації криптографічно строгої ідентифікації віддалених користувачів дозволило виявити реальні залежності між базовими критеріями ефективності ідентифікації: кількістю сеансових паролей, часом навчання, ймовірністю правильної класифікації сеансового пароля і ймовірності хибної класифікації невірного паролю.

ВИСНОВКИ

Як наслідок проведення комплексу теоретичних досліджень, розробок способі та протоколів використання багатошарового персептрона для криптографічно строгої ідентифікації віддалених абонентів, та засобів для їх реалізації на практиці одержані такі результати:

1.Головним недоліком ідентифікації, яка базується на паролях є недостатній рівень захисту від спроб здійснення несанкціонованого доступу, різноманіття типів якого в теперішніх умовах різко зростає. Описаний недолік визначений відсутністю в системах ідентифікації, які базуються на паролях, розвинених криптографічних механізмів, які надають можливість сеансової модифікації ідентифікуючої інформації, вразливість від отримання несанкціонованого доступу з боку системи даних, які застосовуються під час ідентифікації, недотриманням одного з головних принципів захисту інформації – присутності лише одного єдиного власника секретних даних.

2.Ідентифікація віддаленого абонента, котра базується на концепції “нульових знань”, є найбільш надійною з теоретичного боку.

3.Методи, котрі реалізують дану концепцію, математично базуються на аналітично нерозв’язній задачі дискретного логарифмування. З цього слідує, що основними операціями обчислення для методів є мультиплікативні операції модулярної арифметики. Щоб обчислювальна складність реалізації була меншою в існуючих методах реалізовані деякі спрощення, наприклад під час застосовування операції модулярного піднесення до квадрату використовуються числа з набагато меншою розрядністю. Подібні спрощення компенсуються застосуванням декількох (від 3-х до 20) сеансів для обміну даними між абонентом та системою, котрі істотно сповільнюють ідентифікацію абонентів. Отже, головний недолік нині існуючих методів реалізації теоретично строгої ідентифікації, котрі базуються на концепції “нульових знань” в тому, що вони мають високу обчислювальній складності їх опорних обчислювальних процедур. Зрозуміло, що швидкість іденти-

фікації обмежується великою обчислювальною складністю даних операцій та, відповідно, з'являється необхідність у зменшенні її.

4.Головним напрямком для підняття ефективності криптографічно строгої ідентифікації є підняття рівня швидкодії, котре можна досягнуте за допомогою: скорочення кількості циклів обміну ідентифікаційною інформацією між абонентом та системою, застосовування інакшого математичного базису незворотних перетворень так, щоб виконання обчислень мало б не таку велику обчислювальну складність і здійснювало прискорення обрахунків або полегшення схеми під час апаратної реалізації.

5.Показано, що багатошарові персептрони знаходять використання в криптографічному захисті інформації в якості умовно незворотних перетворювачів. Доведено, що рівень захищеності при цьому визначається об’ємом ресурсів, які потрібно витратити, щоб реконструювати по заданим ваговим коефіцієнтам персептрону вхідні сигнали, які формують на виході персептрона позитивний вихідний сигнал класифікації.

6.Показано, що ефективність використання персептрона для криптографічно строгої ідентифікації віддалених користувачів визначається здатністю персептрона вірно класифікувати різні набори вхідних даних, при цьому критеріями ефективності виступають кількість наборів вхідних даних, які класифікуються персептроном як правильні, ймовірність правильної класифікації та ймовірність хибної класифікації.

7.Використання багатошарового персептрону для реалізації криптографічно строгої ідентифікації віддалених користувачів з огляду на жорсткі вимоги до оперативності ідентифікації доцільно лише за умови апаратної реалізації персептрона на серійних цифрових та аналого-цифрових НВІМС або на програмованих матрицях.

8.Запропоновано метод застосування персептрона для криптографічно строгої ідентифікації віддалених користувачів, який включає процедуру навчання, реєстрації та циклу ідентифікації, який відрізняється

вдосконаленою процедурою скалярного множення, за рахунок чого досягається прискорення процесу ідентифікації та також збільшення можливих сеансових паролів.

9.Розроблені програмні засоби моделювання використання багатошарового персептрона для криптографічно строгої ідентифікації віддалених користувачів, в якому персептрона використовується в якості кластерного розпізнавача групи вхідних кодів, які асоціюються з сеансовими ключами користувача.

10.Теоретичне дослідження ефективності розробленого методу показало, що час затрачений на здійснення базової операції схеми ідентифікації, котра базується на моделі персептрона - скалярного множення матриці на вектор під час застосовування запропонованого способу її час затрачений на здійснення зменшується в 21 раз (β=21.3) у порівнянні з традиційним способом її реалізації. Ще одна перевага розробленого запропонованого способу полягає в скороченні витрат ресурсів пам'яті приблизно на 40%.

11.Експериментальні дослідження запропонованого методу показали, що при тривалому навчанні та використанні багатошарових персептронів ймовірність правильної класифікації сеансового паролю користувача доволі висока і становить 0.98 при кількості сеансових паролів 200. При цьому ймовірність хибної класифікації випадкового пароля не перевищує 0.015.

СПИСОК ЛІТЕРАТУРИ

1. Иванов М.А., Криптографические методы защиты информации в компьютерных системах и сетях. М.: ”Кудиз-образ”, 2001.- 368 с.

2. Захариудакис Лефтерис. Метод быстрой аутентификации удаленных пользователей на основе концепции “нулевых знаний” // Наукові записки Українського науково-дослідного інституту зв’язку. 2017.- № 1 (45).– С.109-117.

3. Feige U. Zero knowledge proofs of identity/ U. Feige, A. Fiat A.,Shamir //Journal of Cryptology,- v.1,- n.2- 1988,- P.77-94.

4. Bellekens X. Cyber-PhysicalSecurity Model for Safety-Critical IoT Infrastructures /X. Bellekens, A. Seeam, K. Nieradzinska, C. Tachtatzis, A. Cleary, R. Atkinson, and I. Andonovic // Proceeding Wireless World Research Forum Meeting 35 (WWRF35), Copenhagen, Danemark, 2015.- P.114-118.

5. Ияд Мохд Маджид Ахмад Шахрури, Магрело В.Д. Потапенко М.М. Ідентифікація віддалених абонентів на основі технології нейронних мереж // Матеріали X Міжнародної науково-технічної конференції ”Системний аналіз та інформаційні технології”.-К.:НТУУ ”КПІ”.-2008.- С.357.

6. Wang W. On the Relation Between Identifiability , differential Privacy and Mutual Informational Privacy / Wang W., Ying I., Zhang J. // IEEE Trans. Inf. Theory.- Vol. 62.- 2016.- № 9.- P. 5018-5029.

7. Марковський О.П. Метод строгої ідентифікації віддалених абонентів на основі стандартизованих шифроблоків та хеш-перетворень / О.П. Марковський, Захаріудакіс Лефтеріс., М.Ф. Федотов // Вісник Національного технічного університету України “КПІ” Інформатика, управління та обчислювальна техніка. К.: ТОО „ВЕК+”. 2016.- № 64.- С. 161-165.

8. Марковский А.П. Организация идентификации абонентов многопользовательских систем / Марковский А.П., Азаде Герами, Ияд Мохд Маджид Ахмад Шахрури. // Труды 9-й ”Международной конференции ”Современные информационные и электронные технологии”. Одесса.-2008.- С.129.

9. Марковский А.П. Получение булевых преобразований специальных классов для построения эффективных алгоритмов защиты информации / А.П. Марковский, А.А. Зюзя, В.Д. Шерстюк // Вісник Національного технічного університету України ”КПІ”. Інформатика, управління та обчислювальна техніка. К.,"ВЕК++",- 2008.- № 49.- С.7-13.

10. Шнайер Б. Прикладная криптография. Протоколы, алгоритмы, исходные тексты на языке Си. М.:”Триумф”. - 2002.-816 с.

11. Kittichokenai K. Secret Key-based Identification and Authentication with a Privacy Constraint / K. Kittichokenai, G. Caire. // IEEE Trans. Inf. Theory.- Vol. 62.- 2016.- № 11.- P. 6189-6203.

12. Харин Ю.С., Берник В.И., Матвеев Г.В. Математические основы криптологии / Ю.С. Харин, В.И. Берник, Г.В. Матвеев //.-Минск.: Изд-во БГУ, 1999.- 319 с.

13. Murukesh C. An Efficient Gait Recognition System Based on PCA and Multi-Layer Perceptron / C.Murukesh, K.Thanushkodi // Life Science Journal. – 2013.- Vol.10.- P.1024-1029.

14. Марковський О. П. Використання алгебри полів Галуа для реалізації концепції нульових знань при ідентифікації та автентифікації віддалених користувачів /О.П.Марковський, Захаріудакіс Лефтеріс., В.Р.Максимук // Электронное моделирование. 2017.- № 6.- С.96-110.

15. Марковський О.П. Метод строгої ідентифікації абонентів в телекомунікаційних системах / О.П. Марковський, Лефтеріс Захаріудакіс, М.Ф.

Федотов // XI Міжнародна науково-технічна конференція “Проблеми телекомунікації” ПТ-2017: Збірник матеріалів конференції.К.:КПІ ім. Ігоря Сікорського, 2017.- С.334-337.

16. Bengio S. Secure implementation of identification system / Bengio S., Brassard G., Desmedt Y.G. Goutier C.,Quisquater J.J. // Jornal of Cryptology, v.4, n.3, 1991.- P.186-192.

17. Мухін В.Є. Метод ідентифікації віддалених абонентів на основі концепції “нульових знань” / В.Є. Мухін, Лефтеріс Захаріудакіс, Ю.Н. Герасименко, М.С. Козерацький // Телекомунікаційні та інформаційні технології, 2017 –№1.– С.50-57. рг.-2005.- 286 с.

18. Wright J. Robust recognition via sparse representation / J. Wright, A.Y. Yang, A. Ganesh, S.S. Sastry, and Y. Ma // IEEE Transactions on Pattern Analysis and Machine Intelligence.- 2008,- vol. 31,- no. 2, - P. 210–227.

19. Самофалов К.Г. Повышение эффективности идентификации удаленных абонентов многопользовательских систем / Самофалов К.Г., Тубольцев А.А., Ияд Мохд Маджид Ахмад Шахрури // Проблеми інформатизації та управління. Збірник наукових праць: Випуск 3(14).- К.,НАУ.- 2008.- С.129-136.

20. Ranzato M. Sparse Feature Learning for Deep Belief Networks / M. Ranzato, Y. Boureau, and Y. LeCun, // Advances in neural information processing systems. – 2007.- Vol. 20, P.1185–1192.

21. Demin V.A. Hardware elementary perceptron based on polyaniline memristive devices // V.A. Demin, V.V. Erokhin? A.V. Emelyanov, S. Battistoni, G. Baldi, S. Iannotta // Organic electronic.- 2015.- Vol.25.- P.16-24.

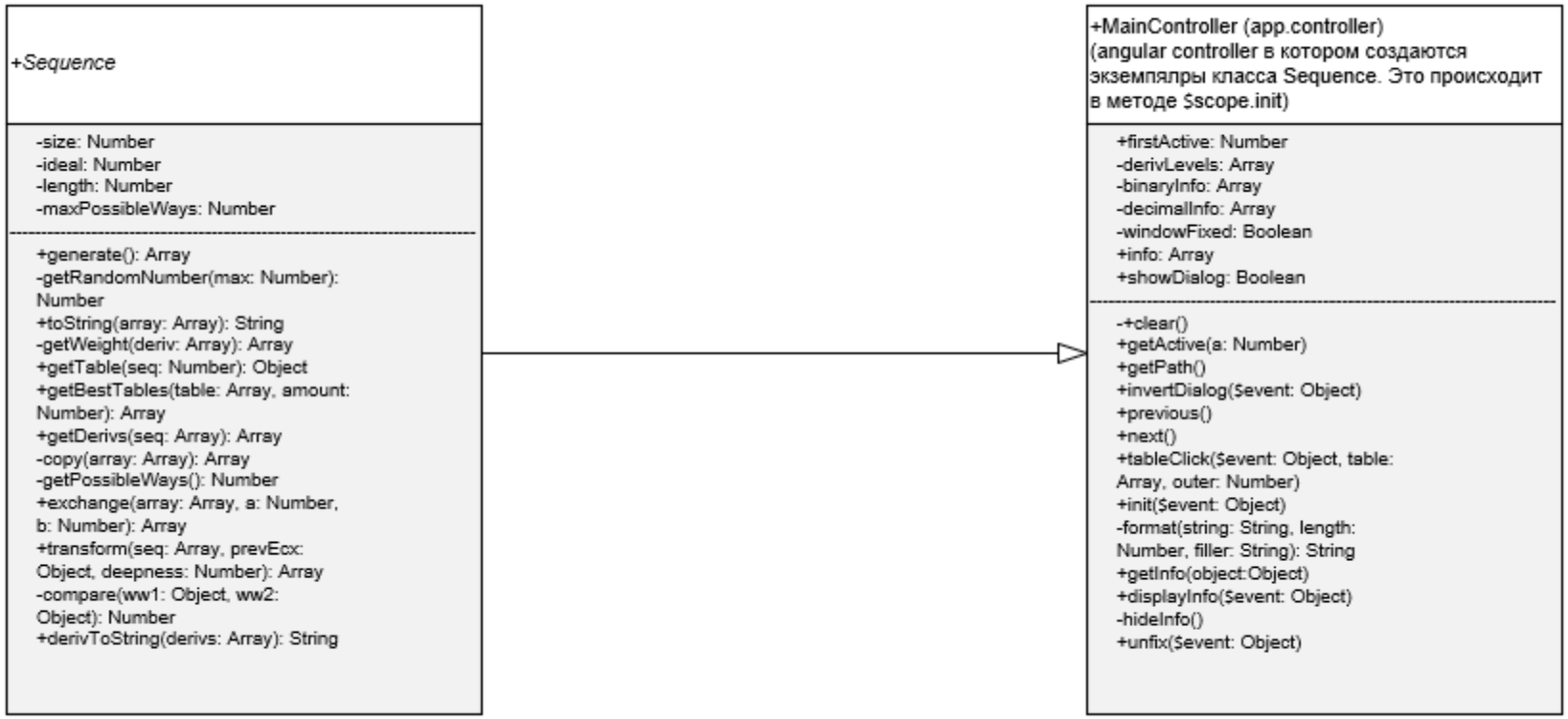
ДОДАТОК А

Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації

Схема ресурсів системи

Аркушів 1

Київ – 2020 року



					ДП 4682.04.000 Д1
Змн	Арк	№ докум	Підпис	Дата	
Розробила		Можаровська С.А.			Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації Схема ресурсів системи
Перевірив		Марковський О.П.			
Н. Контр		Симоненко В.П.			
Затверд.					

Літ	Аркуш	Аркушів
	1	1

НТУУ «КПІ» ФІОТ
ГРУПА ІО-63

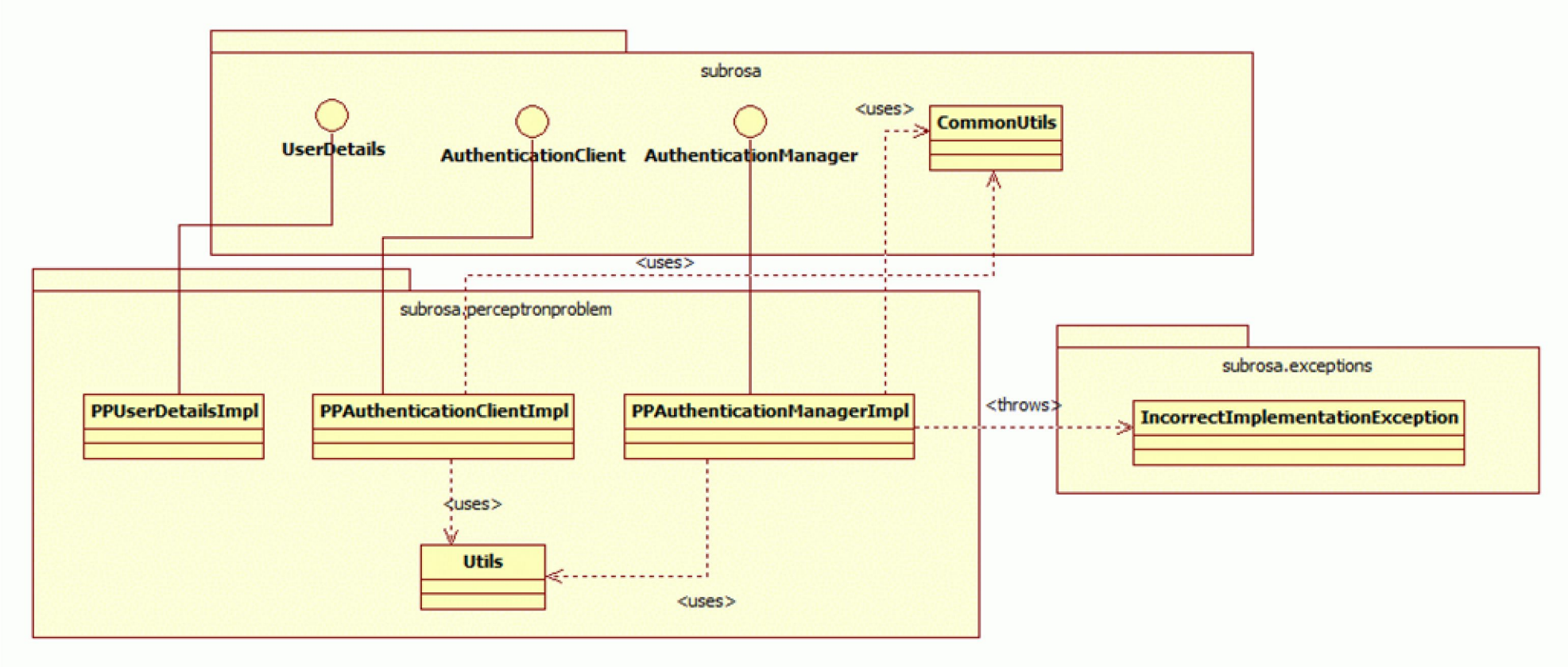
ДОДАТОК Б

Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації

Діаграма класів

Аркушів 1

Київ – 2020 року



					ДП 4682.05.000 Д2	
Змн	Арк	№ докум	Підпис	Дата		
Розробила		Можаровська С.А.			Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації Діаграма класів	Літ / Аркуш 1 / Аркушів 1
Перевірив		Марковський О.П.				
Н. Контр		Симоненко В.П.				НТУУ «КПІ» ФІОТ ГРУПА ІО-63
Затверд.						

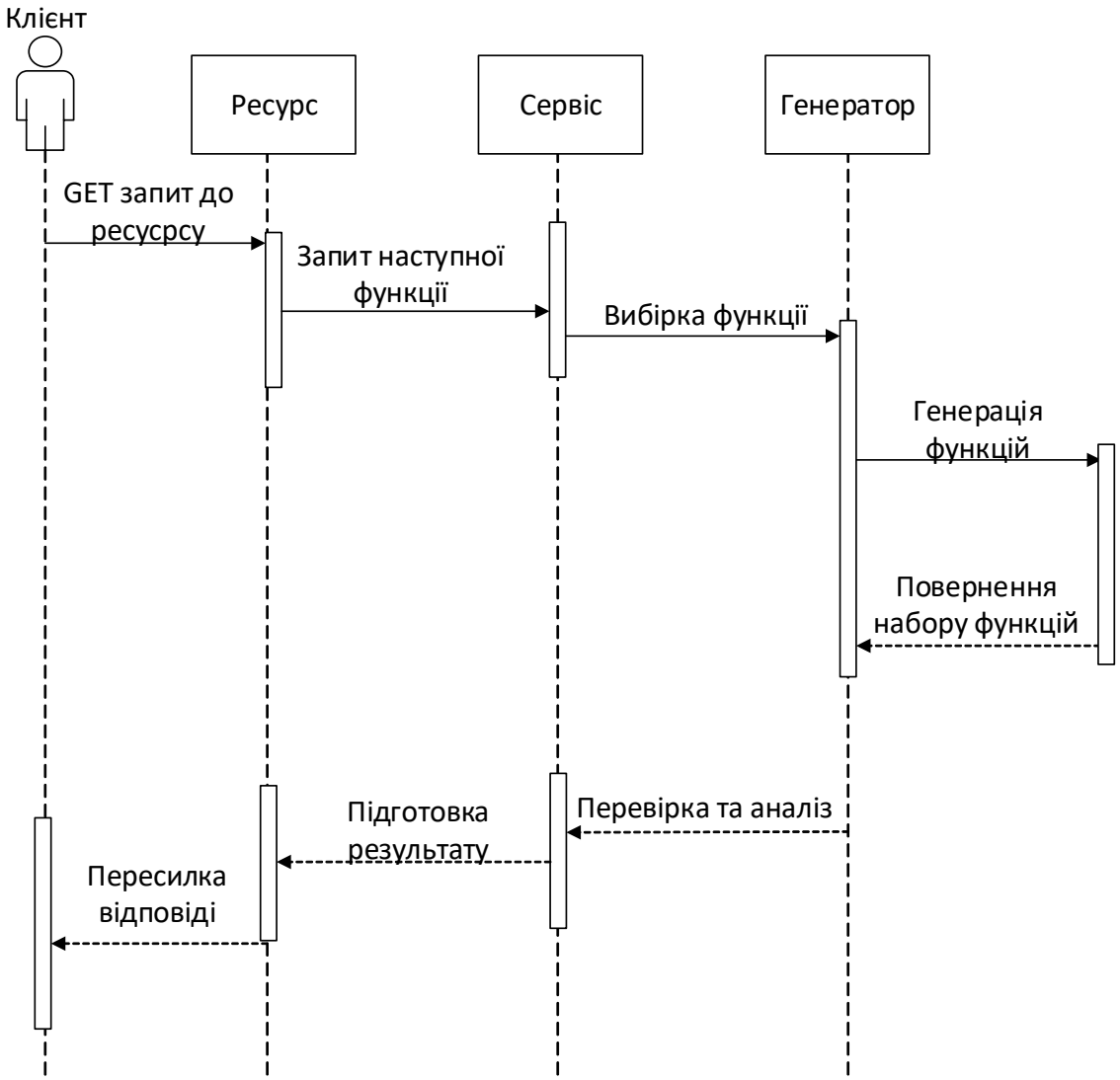
ДОДАТОК В

Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації

Схема роботи програми

Аркушів 1

Київ – 2020 року



					ДП 4682.06.000 Д3		
Зм.	Арк.	№ докум.	Підпис	Дата			
Розробила		Можаровська С.А.			Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації Схема роботи програми	Літ.	Аркуш 1 / Аркушів 1
Перевір.		Марковський О.П.					
Н. контр.		Сімоненко В. П.				НТУУ “КПІ”, ФІОТ, каф. От, гр. ІО-63	
Затверд.							

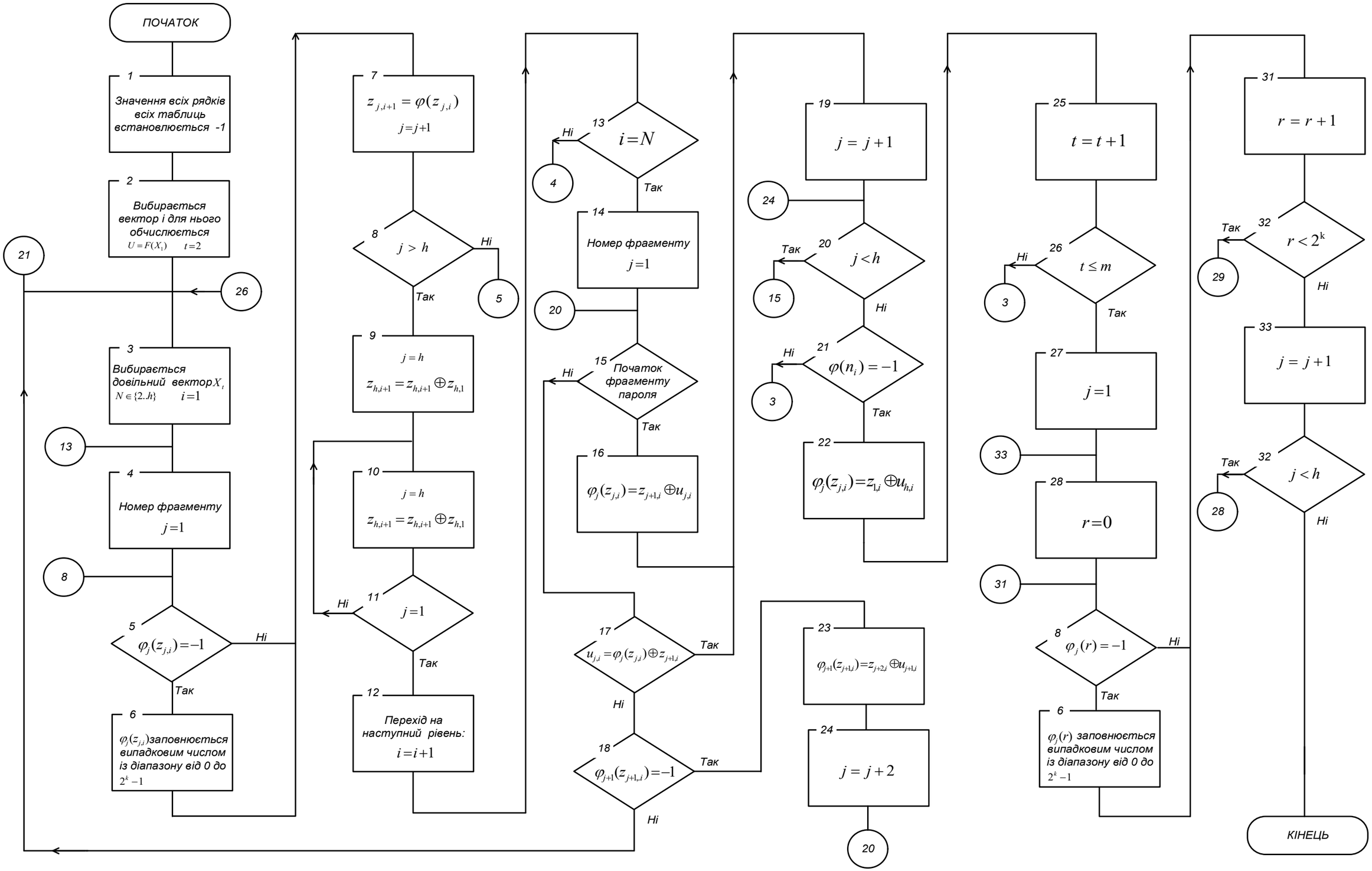
ДОДАТОК Г

Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації

Схема алгоритму

Аркушів 1

Київ – 2020 року



					ДП 4682.07.000 Д4
Змн	Арк	№ докум	Підпис	Дата	
Розробила		Можаровська С.А.			Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації Схема алгоритму
Перевірив		Марковський О.П.			
Н. Контр		Симоненко В.П.			
Затверд.					

Літ	Аркуш	Аркушів
	1	1

НТУУ «КПІ» ФІОТ
ГРУПА ІО-63

ДОДАТОК Д

Програмні засоби моделювання застосування нейронних мереж в системах захисту інформації

Лістинг реалізації ідентифікації

Аркушів 18

Київ – 2020 року

```
package subrosa;
import org.apache.log4j.Logger;

import subrosa.perceptronproblem.PPAuthenticationClientImpl;
import subrosa.perceptronproblem.PPAuthenticationManagerImpl;

/**
 * Main starter.
 */
public class Starter {

  /** Logger. */
  private static Logger log = Logger.getLogger(Starter.class);

  /**
   * Starts the application.
   * @param args arguments
   */
  public static void main(final String[] args) {
    AuthenticationManager am = new PPAuthenticationManagerImpl();
    UserDetails ud = am.createNewAccount();

    AuthenticationClient ac = new PPAuthenticationClientImpl();
    ac.setUserDetails(ud);
    int[] data = ac.formDataToSend();

    log.info("VERIFIED: " + am.verify(data, ac));
  }

}

package subrosa;

/**
 * User details have to contain public and secret keys.
 */
public interface UserDetails {
  /**
   * @return public key
   */
  String getPublicKey();

  /**
   * @return secrete key
   */
  String getSecretKey();

  /**
   * Sets the public key.
   * @param value value to set
   */
  void setPublicKey(final String value);

  /**
   * Sets the secret key.
   * @param value value to set
   */
  void setSecretKey(final String value);
}

package subrosa;
```

```
    /**
     * General interface for the authentication client.
     */
    public interface AuthenticationClient {

      /**
       * @param userDetails user details to set
       */
      void setUserDetails(final UserDetails userDetails);

      int[] formDataToSend();
    }
    package subrosa;

    /**
     * General interface of the authentication manager.
     */
    public interface AuthenticationManager {

      /**
       * @return user details for the new account
       */
      UserDetails createNewAccount();

      /**
       * @param client client to verify
       * @return <code>true</code> in the case of the successful
verification
       */
      boolean verify(final int[] retrievedData, final AuthenticationClient
client);
    }

    /**
     *
     */
    package subrosa;

    import java.util.Random;

    /**
     * Common utility methods.
     */
    public class CommonUtils {

      /** Hidden constructor. */
      private CommonUtils() { }

      /** Random generator. */
      private static Random rGen = new Random();

      public static String getRandomString(final int length) {
        StringBuilder res = new StringBuilder();
        final int cByte = 256;
        for (int i = 0; i < length; i++) {
          byte b = (byte)rGen.nextInt(cByte);
          res.append((char)b);
        }
        return res.toString();
      }

      public static int[] vectorMatrixMul(final int[][] m, final int[] v) {
```

```
        int size = m.length;
        int vsize = v.length;
        assert vsize == m[0].length;

        int[] res = new int[size];
        for (int i = 0; i < size; i++) {
          res[i] = 0;
          for (int j = 0; j < vsize; j++) {
            res[i] += m[i][j] * v[j];
          }
        }
        return res;
      }

      public static int[] addVectors(final int[] a, final int b[]) {
        int[] res = new int[a.length];
        for (int i = 0; i < b.length; i++) {
          res[i] = a[i] + b[i];
        }
        return res;
      }
    }

    /**
     *
     */
    package subrosa.exceptions;

    /**
     * Throws in the case of using the incorrect implementation of the
authentication interfaces.
     */
    public class IncorrectImplementationException extends RuntimeException {

      /** serialVersionUID. */
      private static final long serialVersionUID = 4949624804182281939L;

      /**
       * Constructor.
       * @param message message
       */
      public IncorrectImplementationException(final String message) {
        super(message);
      }

      /**
       * Constructor.
       * @param message message
       * @param e cause
       */
      public IncorrectImplementationException(final String message, final
Throwable e) {
        super(message, e);
      }

      /** Constructor. */
      public IncorrectImplementationException() {
        super("Incorrect implementaion used!");
      }
    }

    /**
```

```
 *
 */
package subrosa.perceptronproblem;

import java.util.Random;

import subrosa.AuthenticationClient;
import subrosa.CommonUtils;
import subrosa.UserDetails;
import subrosa.exceptions.IncorrectImplementationException;
public class PPAuthenticationClientImpl implements AuthenticationClient
{

  /** User details. */
  private PPUserDetailsImpl userDetails = null;

  private int[] p, q;
  private int[][] a2;
  private int[] v2, w, r;

  private Random rGen = new Random();

  /**
   * {@inheritDoc}
   */
  public void setUserDetails(final UserDetails userDetails) {
    if (!(userDetails instanceof PPUserDetailsImpl)) {
      throw new IncorrectImplementationException("PPUserDetailsImpl is
required!");
    }

    this.userDetails = (PPUserDetailsImpl)userDetails;
    this.p = Utils.permutation(this.userDetails.getPublicA().length);
    this.q = Utils.permutation(this.userDetails.getPublicA()[0].length);
    this.a2 = Utils.permutateMatrix(this.userDetails.getPublicA(), p,
q);
    prepareV2();

    w = new int[this.q.length];
    r = new int[this.q.length];
    for (int i = 0; i < this.q.length; i++) {
      w[i] = rGen.nextInt(w.length);
      r[i] = w[i] + v2[i];
    }
  }

  /**
   * @return the userDetails
   */
  public PPUserDetailsImpl getUserDetails() {
    return userDetails;
  }

  private void prepareV2() {
    final int m = this.userDetails.getSecretV().length;
    this.v2 = new int[m];
    for (int i =0; i < m; i++) {
      v2[i] = this.userDetails.getSecretV()[q[i]];
    }
  }

  /**
```

```
     * @return the p
     */
    public int[] getP() {
      return p;
    }

    /**
     * @return the q
     */
    public int[] getQ() {
      return q;
    }

    /**
     * @return the a2
     */
    public int[][] getA2() {
      return a2;
    }

    /**
     * @return the v2
     */
    public int[] getV2() {
      return v2;
    }

    /**
     * @return the w
     */
    public int[] getW() {
      return w;
    }

    /**
     * @return the r
     */
    public int[] getR() {
      return r;
    }

    private int getH0() {
      return Utils.vectorHash(p);
    }
    private int getH1() {
      return Utils.vectorHash(w);
    }
    private int getH2() {
      return Utils.vectorHash(r);
    }
    private int getH3() {
      return Utils.vectorHash(CommonUtils.vectorMatrixMul(a2, w));
    }
    private int getH4() {
      return Utils.vectorHash(CommonUtils.vectorMatrixMul(a2, r));
    }

    public int[] formDataToSend() {
      int[] result = {getH0(), getH1(), getH2(), getH3(), getH4()};
      return result;
    }
  }
```

```
package subrosa.perceptronproblem;

import java.util.Arrays;
import java.util.HashSet;
import java.util.Random;
import java.util.Set;

import org.apache.log4j.Logger;

import subrosa.AuthenticationClient;
import subrosa.AuthenticationManager;
import subrosa.CommonUtils;
import subrosa.UserDetails;
import subrosa.exceptions.IncorrectImplementationException;

/**
 * {@link AuthenticationManager} implementation for the perceptron
problem.
 */
public class PPAuthenticationManagerImpl implements
AuthenticationManager {

  /** Logger. */
  private static Logger log =
Logger.getLogger(PPAuthenticationManagerImpl.class);

  /** Sizes of e-vectors and matrixes. */
  private static final int SIZE_M = 101, SIZE_N = 117;

  private Random rGen = new Random();

  /**
   * {@inheritDoc}
   */
  public UserDetails createNewAccount() {
    log.info("Creating the new account");
    int[] v = Utils.randomEVector(SIZE_N);
    int[][] a = Utils.randomEMatrix(SIZE_M, SIZE_N);
    if (log.isDebugEnabled()) {
      log.debug("v: " + Arrays.toString(v));
    }
    for (int i = 0; i < SIZE_M; i++) {
      int av = 0;
      for (int j = 0; j < SIZE_N; j++) { av += v[j] * a[i][j]; }
      if (av < 0) {
        for (int j = 0; j < SIZE_N; j++) { a[i][j] = -a[i][j]; }
      }
      if (log.isDebugEnabled()) {
        log.debug("av[" + i + "] = " + av);
        log.debug("a[" + i + "] = " + Arrays.toString(a[i]));
      }
    }

    Set<Integer> s = new HashSet<Integer>();
    for (int i = 0; i < SIZE_M; i++) {
      int av = 0;
      for (int j = 0; j < SIZE_N; j++) { av += v[j] * a[i][j]; }
      s.add(av);
    }
    if (log.isDebugEnabled()) {
      log.debug("S: " + s);
```

```
        }

        PPUserDetailsImpl result = new PPUserDetailsImpl();
        result.setPublicA(a);
        result.setPublicS(s);
        result.setSecretV(v);
        return result;
      }

      /**
       * {@inheritDoc}
       */
      public boolean verify(final int[] retrievedData, final
AuthenticationClient client) {
        if (!(client instanceof PPAuthenticationClientImpl)) {
          throw new
IncorrectImplementationException("PPAuthenticationClientImpl is required.");
        }

        PPAuthenticationClientImpl cl = (PPAuthenticationClientImpl)client;
        int way = rGen.nextInt(4);
        log.debug("way: " + way);
        way = 0;
        switch (way) {
        case 0:
          if (retrievedData[0] != Utils.vectorHash(cl.getP())) {
log.error("Bad h0"); return false; }
          if (retrievedData[1] != Utils.vectorHash(cl.getW())) {
log.error("Bad h1"); return false; }
          int[] paqw =
CommonUtils.vectorMatrixMul(Utils.permutateMatrix(cl.getUserDetails().getPub
licA(), cl.getP(), cl.getQ()), cl.getW());
          if (retrievedData[3] != Utils.vectorHash(paqw)) { log.error("Bad
h3"); return false; }
          return true;
        case 1:
          if (retrievedData[0] != Utils.vectorHash(cl.getP())) {
log.error("Bad h0"); return false; }
          if (retrievedData[2] != Utils.vectorHash(cl.getR())) {
log.error("Bad h2"); return false; }
          int[] paqr =
CommonUtils.vectorMatrixMul(Utils.permutateMatrix(cl.getUserDetails().getPub
licA(), cl.getP(), cl.getQ()), cl.getR());
          if (retrievedData[4] != Utils.vectorHash(paqr)) { log.error("Bad
h04"); return false; }
          return true;
        case 2:
          int[] a2w = CommonUtils.vectorMatrixMul(cl.getA2(), cl.getW());
          if (retrievedData[3] != Utils.vectorHash(a2w)) { log.error("Bad
h3"); return false; }
          int[] a2v = CommonUtils.vectorMatrixMul(cl.getA2(), cl.getV2());
          if (retrievedData[4] !=
Utils.vectorHash(CommonUtils.addVectors(a2w, a2v))) { log.error("Bad h4");
return false; }
          boolean allInS = true;
          for (int x : a2v) {
            if (!(allInS = cl.getUserDetails().getPublicS().contains(x))) {
break; }
          }
          return allInS;
        case 3:
```

```
          if (retrievedData[1] != Utils.vectorHash(cl.getW()))
{log.error("Bad h01");  return false; }
          if (retrievedData[2] !=
Utils.vectorHash(CommonUtils.addVectors(cl.getW(), cl.getV2()))) { return
false; }
          boolean allInD = true;
          for (int x : cl.getV2()) {
            if (!(allInD = (x == 1) || (x == -1))) { break; }
          }
          return allInD;
        }
        return false;
      }



    }

    /**
     *
     */
    package subrosa.perceptronproblem;

    import java.util.Set;

    import subrosa.CommonUtils;
    import subrosa.UserDetails;

    /**
     * {@link UserDetails} implementation for the perceptron problem.
     */
    class PPUserDetailsImpl implements UserDetails {

      /** Offset. */
      private static final int OFFSET = 5;
      /** Random part length. */
      private static final int RANDOM_LENGTH = 2;

      /** E-matrix A. */
      private int[][] publicA;
      /** Set of nonnegative values. */
      private Set<Integer> publicS;

      /** E-vector V. */
      private int[] secretV;

      /**
       * {@inheritDoc}
       */
      public String getPublicKey() {
        int cSum = 0;
        final int cByte = 256;
        StringBuilder res = new
StringBuilder(CommonUtils.getRandomString(RANDOM_LENGTH));
        for (int x : publicS) {
          cSum += x;
          byte b = (byte)(x + OFFSET);
          res.append((char)b);
          res.append(CommonUtils.getRandomString(RANDOM_LENGTH));
        }
        for (int[] xx : publicA) {
          for (int x : xx) {
```

```
            cSum += x;
            byte b = (byte)(x + OFFSET);
            res.append((char)b);
            res.append(CommonUtils.getRandomString(RANDOM_LENGTH));
          }
        }
        byte b = (byte)(cSum % cByte);
        res.append((char)b);
        return res.toString();
      }

      /**
       * {@inheritDoc}
       */
      public String getSecretKey() {
        int cSum = 0;
        final int cByte = 256;
        StringBuilder res = new
StringBuilder(CommonUtils.getRandomString(RANDOM_LENGTH));
        for (int x : secretV) {
          cSum += x;
          byte b = (byte)(x + OFFSET);
          res.append((char)b);
          res.append(CommonUtils.getRandomString(RANDOM_LENGTH));
        }
        byte b = (byte)(cSum % cByte);
        res.append((char)b);
        return res.toString();
      }

      /**
       * @return the publicA
       */
      public int[][] getPublicA() {
        return publicA;
      }

      /**
       * @return the publicS
       */
      public Set<Integer> getPublicS() {
        return publicS;
      }

      /**
       * @return the secretV
       */
      public int[] getSecretV() {
        return secretV;
      }

      /**
       * @param publicA the publicA to set
       */
      public void setPublicA(final int[][] publicA) {
        this.publicA = publicA;
      }

      /**
       * @param publicS the publicS to set
       */
      public void setPublicS(final Set<Integer> publicS) {
```

```
    this.publicS = publicS;
  }

  /**
   * @param secretV the secretV to set
   */
  public void setSecretV(final int[] secretV) {
    this.secretV = secretV;
  }

  /**
   * {@inheritDoc}
   */
  public void setPublicKey(final String value) {
    // TODO Parsing public key

  }

  /**
   * {@inheritDoc}
   */
  public void setSecretKey(final String value) {
    // TODO Parsing secrete key

  }

}

package subrosa.perceptronproblem;

import java.util.ArrayList;
import java.util.Random;

/**
 * Utility methods.
 */
final class Utils {

  /** Hidden constructor. */
  private Utils() {}

  /** Random generator. */
  private static Random rGen = new Random();

  /**
   * @return random value belonging <code>{-1;1}</code>
   */
  private static int randomEValue() {
    return (rGen.nextBoolean()) ? 1 : -1;
  }

  /**
   * @param size vector's size
   * @return the random e-vector
   */
  public static int[] randomEVector(final int size) {
    int[] res = new int[size];
    for (int i = 0; i < size; i++) {
      res[i] = randomEValue();
    }
    return res;
  }
```

```
        /**
         * @param m count of rows
         * @param n count of columns
         * @return the random e-matrix
         */
        public static int[][] randomEMatrix(final int m, final int n) {
          int[][] res = new int[m][n];
          for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++) {
              res[i][j] = randomEValue();
            }
          }
          return res;
        }

        /**
         * @param vector the vector
         * @return hash value for the vector
         */
        public static int vectorHash(final int[] vector) {
          int res = 1;
          final int c5 = 5;
          for (int x : vector) {
            int r = res;
            res <<= c5;
            res -= r;
            res += x;
          }
          return res;
        }

        /**
         * @param m margin value of the set
         * @return permutation over <code>{0, ..., m - 1}</code>
         */
        public static int[] permutation(final int m) {
          ArrayList<Integer> set = new ArrayList<Integer>(m);
          for (int i = 0; i < m; i++) { set.add(i); }
          int[] res = new int[m];
          for (int i = 0; i < m; i++) {
            int index = rGen.nextInt(set.size());
            res[i] = set.remove(index);
          }
          return res;
        }

        private static void swapColumns(final int[][] a, final int i, final
int j) {
          for (int k = 0; k < a.length; k++) {
            int e = a[k][i];
            a[k][i] = a[k][j];
            a[k][j] = e;
          }
        }

        public static int[][] permutateMatrix(final int[][] a, final int[]
rows, final int[] cols) {
          int m = a.length; int n = a[0].length;
          int[][] result = new int[m][n];
          for (int i = 0; i < m ;i++) {
            result[i] = a[rows[i]].clone();
```

```
        }
        for (int i = 0; i < n; i++) {
          swapColumns(result, i, cols[i]);
        }
        return result;
      }
    }
```